Internship Project Report

Implementation Of Serial and Ethernet Communication using Xilinx Vivado, Xilinx SDK and ZYNQ Evaluation Board.

Project Guide: A Sudheer Babu (Scientist E)

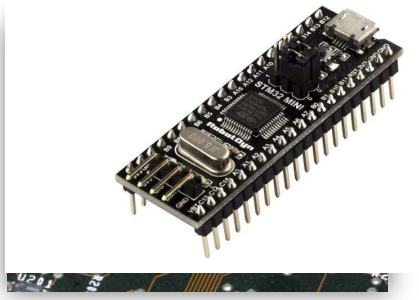
Name: Jaruplavath Uday Karthik (2214053)

SINT(C) (DRDL/RCI)

MICROPROCESSORS:

A microprocessor, at its core, is a highly intricate electronic component central to the operation of digital systems. It's composed of numerous interconnected elements, primarily the central processing unit (CPU), which serves as the brains of the system. The CPU encompasses various components like arithmetic logic units, registers, and a control unit, each playing a critical role in processing instructions and managing data.

To function, microprocessors rely on data buses and address buses to facilitate communication with memory and peripherals. They follow a fetch-decode-execute cycle, retrieving instructions from memory, deciphering their meaning, and then executing the corresponding operations. This process is repeated countless times per second, enabling the rapid computation and manipulation of data.



Microprocessors find widespread use in computers, smartphones, embedded systems, and many other electronic devices. To fabricate these complex circuits, Very Large-Scale Integration (VLSI) techniques are employed, allowing for the integration of thousands to billions of transistors onto a single chip. This level of integration empowers microprocessors to perform a multitude of tasks at incredible speeds, making them fundamental components of modern electronics.

MICROCONTROLLERS:

Microcontrollers are compact, self-contained computing devices designed for embedded systems. They consist of a central processing unit (CPU), memory, input/output peripherals, and often some form of programmable interface. Unlike general-purpose microprocessors, microcontrollers are tailored for specific tasks, making them ideal for controlling a range of electronic devices and systems. They're commonly found in everyday items like appliances, automotive systems, industrial machinery, and consumer electronics.

Microcontrollers execute programmed instructions to interact with sensors, process data, and control connected hardware. Their design often includes digital and analog input/output pins, timers, communication interfaces (e.g., UART, SPI, I2C), and memory (flash, RAM,

EEPROM) for program storage and data handling. Users can write software code (firmware) to define the microcontroller's functionality, making it versatile and adaptable.

Microcontrollers come in various architectures, from simple 8-bit designs to more advanced 32-bit versions, with the choice depending on the application's complexity and power requirements. They're vital in automating and controlling various processes, making them a foundational technology in the world of embedded systems.

DIFFERENCE BETWEEN A MIDCORPROCESSOR AND MICROCONTROLLER:

- **Purpose:**

  - **Microprocessor:** Primarily designed for general-purpose computing tasks and executing complex software applications, typically used in computers and servers.
  - **Microcontroller:** Tailored for specific control and automation applications, focusing on managing hardware components in embedded systems.

- **Components:**

  - **Microprocessor:** Typically consists of a CPU, control unit, and memory, often requires external peripherals for specific tasks.
  - **Microcontroller:** Contains a CPU, memory, and integrated input/output peripherals like GPIO pins, timers, and communication interfaces on a single chip.

- **Complexity:**

  - **Microprocessor:** Generally more complex and powerful, capable of handling diverse and demanding tasks.
  - **Microcontroller:** Less complex, optimized for simpler, specific tasks, offering efficiency and cost-effectiveness.

- **System Integration:**

  - **Microprocessor:** Requires external components and additional chips for interfacing with the outside world, making it suitable for systems with varied requirements.
  - **Microcontroller:** Designed for self-contained, integrated systems, simplifying circuit design and reducing component count.

- **Power Consumption:**

  - **Microprocessor:** Consumes more power, often requiring active cooling systems in high-performance applications.
  - **Microcontroller:** Typically operates with lower power consumption, ideal for battery-powered or energy-efficient devices.

- **Application:**

  - **Microprocessor:** Suited for desktop computers, laptops, servers, and high-performance computing devices.
  - **Microcontroller:** Commonly used in embedded systems, such as IoT devices, robotics, automotive control systems, and home appliances.

- **Cost:**

  - **Microprocessor:** Often more expensive due to its higher processing power and versatility.
  - **Microcontroller:** Generally cost-effective, making it a preferred choice for mass-produced, cost-sensitive applications.

- **Development Complexity:**

  - **Microprocessor:** Software development can be complex, as it involves writing and optimizing software for a wide range of applications.

- **Microcontroller:** Easier to develop for, with specialized IDEs and libraries tailored to specific applications, simplifying the programming process.

In summary, microprocessors are versatile and powerful, suitable for general-purpose computing, while microcontrollers are efficient, cost-effective solutions designed for dedicated control and automation tasks in embedded systems. The choice between them depends on the specific needs of the application.

ADVANTAGES AND DISADVANTAGES OF MICROPROCESSORS:

ADVANTAGES:

- **Versatility:**
  Microprocessors are versatile and capable of running a wide range of software applications, making them suitable for general-purpose computing tasks.
- **Processing Power:**
  They offer high processing power, which is essential for handling complex and computation-intensive tasks.
- **Broad Compatibility:**
  Microprocessors support various operating systems and software, ensuring compatibility with a wide array of applications.
- **Expandability:**
  They can be paired with external peripherals to meet specific requirements.
- **Multitasking:**
  Microprocessors are proficient at multitasking, allowing them to execute multiple applications simultaneously.

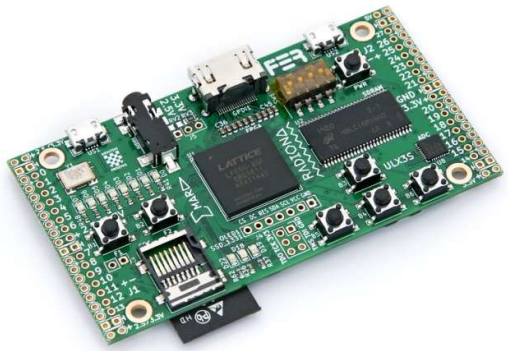
DISADVANTAGES:

- **Cost:** Microprocessors are often more expensive due to their high processing power and flexibility.
- **Power Consumption:** They consume more power, leading to heat generation and the need for cooling systems.
- **Complex Design:** Developing software for microprocessors can be complex and resource-intensive.

- **External Components:** They may require additional components for interfacing with the external world, adding complexity to the system.

ADVANTAGES AND DISADVANTAGES OF MICROCONTROLLERS:

ADVANTAGES:



- **Integration:** Microcontrollers are highly integrated, with the CPU, memory, and peripherals on a single chip, simplifying circuit design.
- **Cost-Effective:** They are cost-effective solutions, making them ideal for mass-produced devices.
- **Low Power Consumption:** Microcontrollers typically operate with lower power consumption, extending battery life in portable devices.
- **Real-Time Control:** They excel at real-time control and automation tasks, offering predictable and timely responses.
- **Easy Development:** Microcontrollers often have specialized development tools and libraries, making software development more accessible for specific applications.

DISADVANTAGES:

- **Limited Processing Power:** Microcontrollers have limited processing power compared to microprocessors, which can be a constraint for complex applications.
- **Application-Specific:** They are designed for specific tasks, making them less suitable for general-purpose computing.
- **Limited Compatibility:** Microcontrollers may have limitations in terms of operating system support and software compatibility.
- **Expandability:** They are less flexible in terms of adding external components and peripherals, limiting their adaptability for diverse tasks.

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.

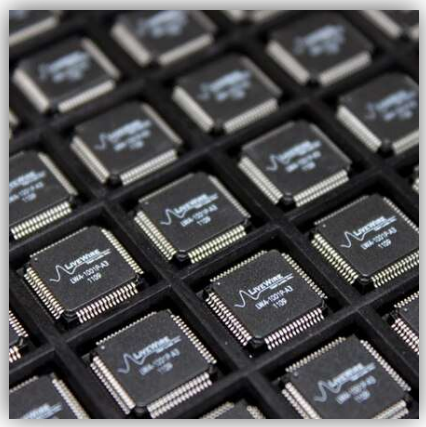
- **Enclustra:** Enclustra focuses on FPGA and SoC modules, targeting a wide range of applications, from embedded systems to high-performance computing.

- **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.

ZC702 EVALUATION BOARD

ZC702 Evaluation Board consists of zynq processing system, user LEDs, user switches, Timers, Ethernet slot, DDR memories, Etc. These FPGA can be programmed based on the requirement by using the software called VIVADO 2015.4. Before working with this board make sure the jumper wire setting i.e SW16 into JTAG mode to dump the program.

These board consists of UART, JTAG, and Ethernet slots pins which are to be connected to the PC:

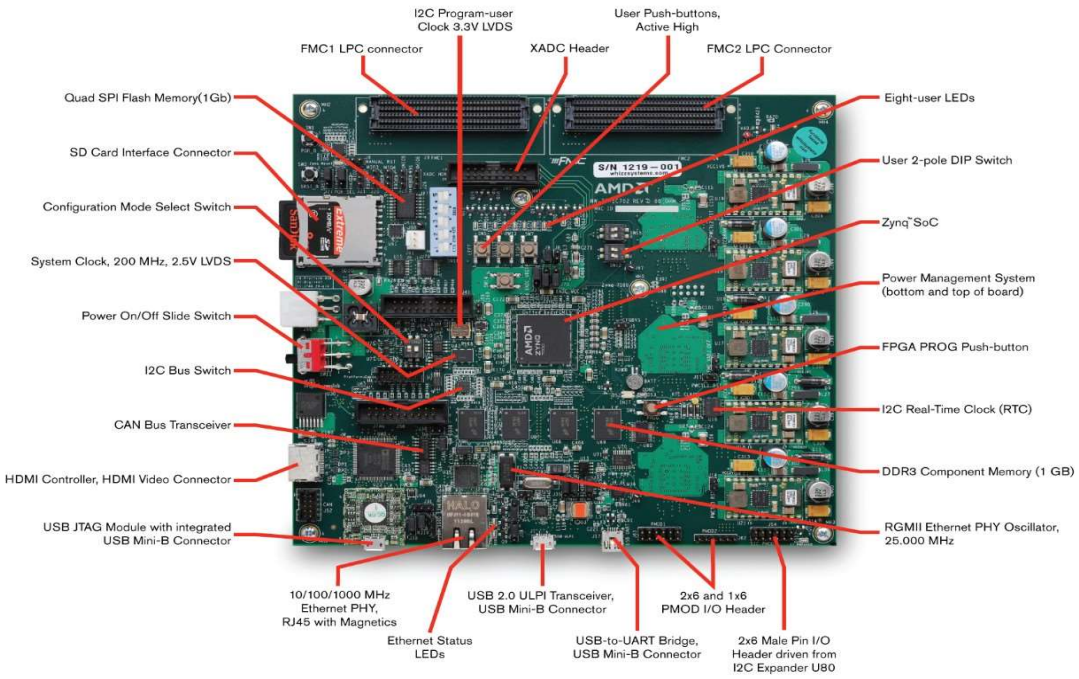
- JTAG – This Pin used to program the board

  Jumper Settings (A blue coloured 5 Switches on the board

  All down (i.e 0)- JTAG Mode

- UART- This pin is connected to the serial monitor (UART Bridge) of the PC.

The ZC702 Evaluation Board is a development platform designed by Xilinx for engineers, researchers, and hobbyists to evaluate and prototype designs using Xilinx's Zynq-7000 family of system-on-chip (SoC) devices. This board is part of the Zynq-7000 All Programmable SoC (AP SoC) family and provides a comprehensive environment for embedded systems development.



The ZC702 Evaluation Board, as a development platform for Xilinx's Zynq-7000 SoC, has several uses, advantages, and disadvantages.

USES:

- **Embedded System Development:** The primary use of the ZC702 board is for embedded system development. It allows engineers and developers to create and prototype custom hardware and software solutions for a wide range of applications.
- **Digital Signal Processing (DSP):** The high-performance FPGA fabric, coupled with dual ARM Cortex-A9 cores, makes it suitable for DSP applications, such as real-time image and video processing, audio processing, and signal filtering.
- **Industrial Control and Automation:** The board can be used in industrial settings for control and automation tasks. It offers various I/O options and interfaces, making it versatile for interfacing with sensors, actuators, and other industrial equipment.
- **Communications and Networking:** The ZC702 board can be used for developing networking and communication applications, including software-defined radio, network protocol processing, and data encryption/decryption.
- **Prototyping and Research:** It serves as a valuable platform for research and prototyping projects where custom hardware accelerators and software components are required.

ADVANTAGES:

1. Integrated Processing System and FPGA:

- **Combines a dual-core ARM Cortex-A9 processor with Xilinx 7-series FPGA logic, offering flexibility for both software and hardware design.**

2. High Performance:

- **Capable of handling high-performance applications due to its powerful processing system and advanced FPGA capabilities.**

3. Comprehensive Connectivity:

- **Provides a wide range of connectivity options, including Gigabit Ethernet, USB, HDMI, SATA, PCIe, and more, making it suitable for diverse applications.**

4. Expandable and Customizable:

- **Features FMC (FPGA Mezzanine Card) and PMOD headers, allowing for expansion and customization with additional modules and peripherals.**

5. Robust Memory Options:

- **Equipped with 1 GB DDR3 SDRAM, 256 MB Quad-SPI Flash, and an 8 GB SD card, supporting complex applications and large data sets.**

6. Comprehensive Development Tools:

- **Compatible with Xilinx's Vivado Design Suite for FPGA design and the Xilinx SDK for software development, providing a complete development environment.**

7. Versatile Application Development:

- **Supports both Linux and bare-metal application development, catering to a wide range of embedded system requirements.**

8. Efficient Debugging and Testing:

- **Includes JTAG, UART, and other debugging interfaces, facilitating efficient debugging and testing of designs.**

9. Prototyping and Evaluation:

- **Ideal for prototyping and evaluating designs, enabling quick validation of concepts before moving to final product development.**

10. Learning and Experimentation:

- **Excellent educational tool for learning about ARM processors, FPGA design, and embedded systems, providing hands-on experience.**

11. Community and Support:

- **Backed by extensive documentation, tutorials, and a strong community of users and developers, offering valuable support and resources.**

12. Time-to-Market:

- **Accelerates the time-to-market for new products by providing a ready-to-use platform for development and testing.**

DISADVANTAGES:

- **Cost:** The ZC702 board, like many development boards from Xilinx, can be relatively expensive, which may not be suitable for hobbyist or small budget projects.
- **Complexity:** Working with FPGAs and complex SoC devices can be challenging, especially for those new to FPGA development. The learning curve can be steep.
- **Power Consumption:** High-performance boards like ZC702 can consume significant power, so power management and cooling may be required in some applications.
- **Size and Form Factor:** The board's size and form factor may not be suitable for space-constrained applications or compact product designs.
- **Compatibility:** Compatibility with third-party hardware and software can sometimes be an issue, as not all components and libraries may be readily available or easily adaptable.

In conclusion, the ZC702 Evaluation Board is a potent platform for embedded system development, offering high performance and versatility. While it provides numerous advantages, including scalability, community support, and the integration of software and hardware, it is not without its drawbacks, such as cost, complexity, and power consumption.

It is a valuable tool for projects that demand the capabilities it offers but may not be the best fit for all applications.

VIVADO 2015.4

It is a software where we can write the program to work on the evaluation board. Internally we have software development kit(SDK) in vivado.

Xilinx software consists of predefined libraries where we can include the files based on the requirement.

There are different versions in the vivado where we can access them through License. We are currently working on version 2015.4

Vivado consists of a set of Libraries were we can connect the different blocks or IP's by choosing the required board we are working for it. These connections of block can be programed the board by using Xilinx(SDK).

There is a sequence to be followed in software to implement the OUTPUT.

XILINX SDK

Xilinx Software Development Kit (SDK) is an integrated development environment (IDE) designed for developing embedded applications on Xilinx platforms. Here's a brief overview of its main features and uses:

Key Features

1. Integrated Development Environment: Provides a comprehensive environment for developing, debugging, and deploying embedded applications.
2. Board Support Packages (BSPs): Offers BSPs for various Xilinx hardware platforms, facilitating easier development and deployment.
3. Application Development: Supports the development of bare-metal and Linux-based applications for Xilinx devices.
4. System Debugging: Includes tools for debugging hardware and software, such as integrated debuggers and performance analysis tools.
5. Peripheral Drivers: Provides a library of peripheral drivers for easy integration of external devices.
6. Automation: Supports automation of build and deployment processes through scripting.

Uses

1. **Embedded Systems**: Used for developing applications on Xilinx embedded systems, such as Zynq-7000 SoCs and MicroBlaze.
2. **FPGA Development**: Supports software development for applications running on FPGAs.
3. **System Design**: Facilitates the design and implementation of complex systems by providing tools for both hardware and software development.
4. **Prototyping**: Enables rapid prototyping and testing of embedded applications.

PROJECT-5

IMPLEMENTING LWIP EHTERNET USING ZYNQ ZC702 FPGA BOARD

Aim: Establishing connection between PC and ZYNQ ZC702 FPGA board using Ethernet cable and transfer data from board to pc and pc to board.

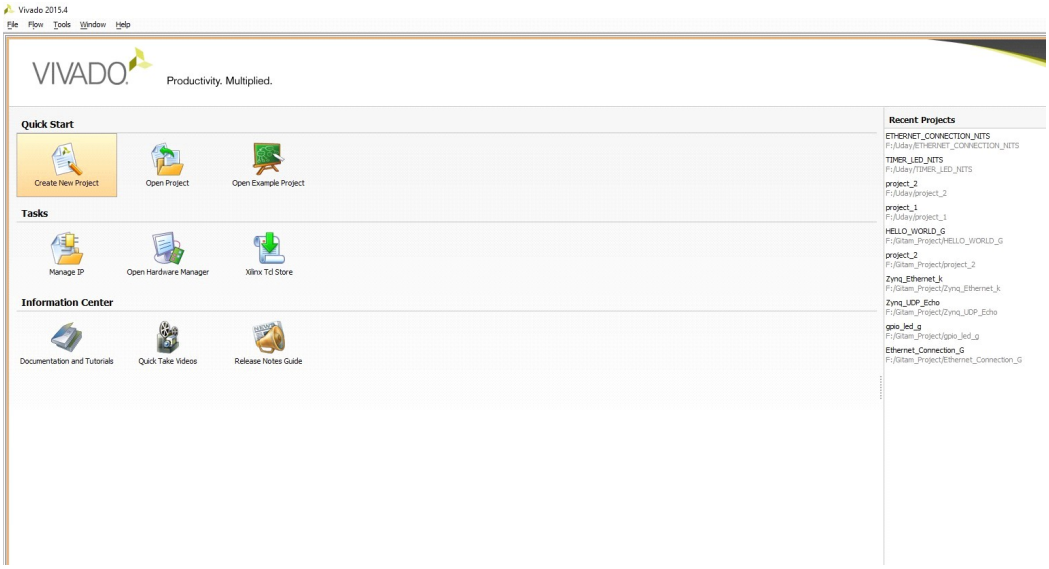
Required Tools: ZYNQ C702 FPGA Board, JTAG cable, UART cable

Software: Xilinx Vivado, Xilinx SDK

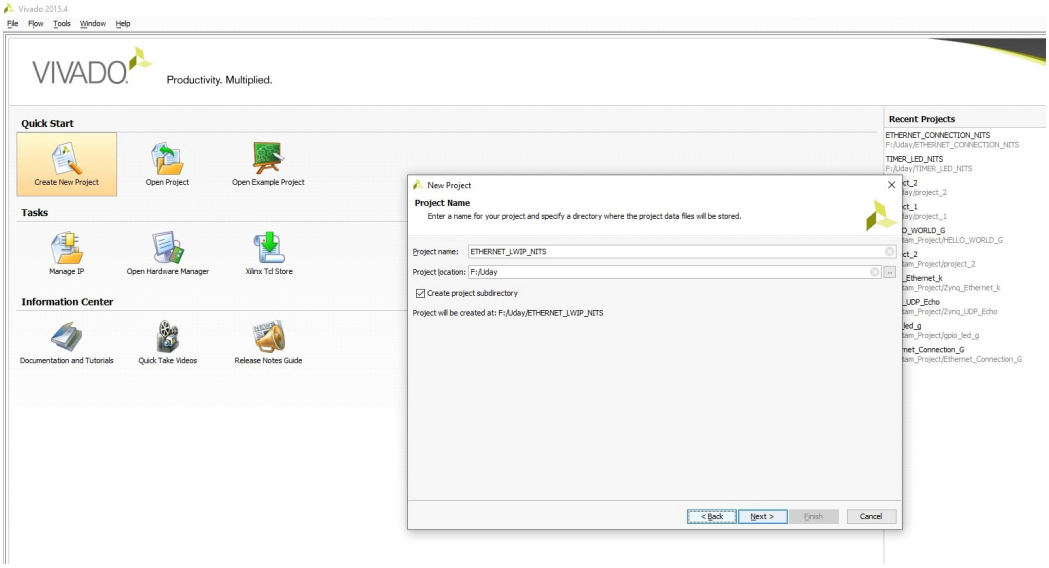
Procedure:

Step 1:

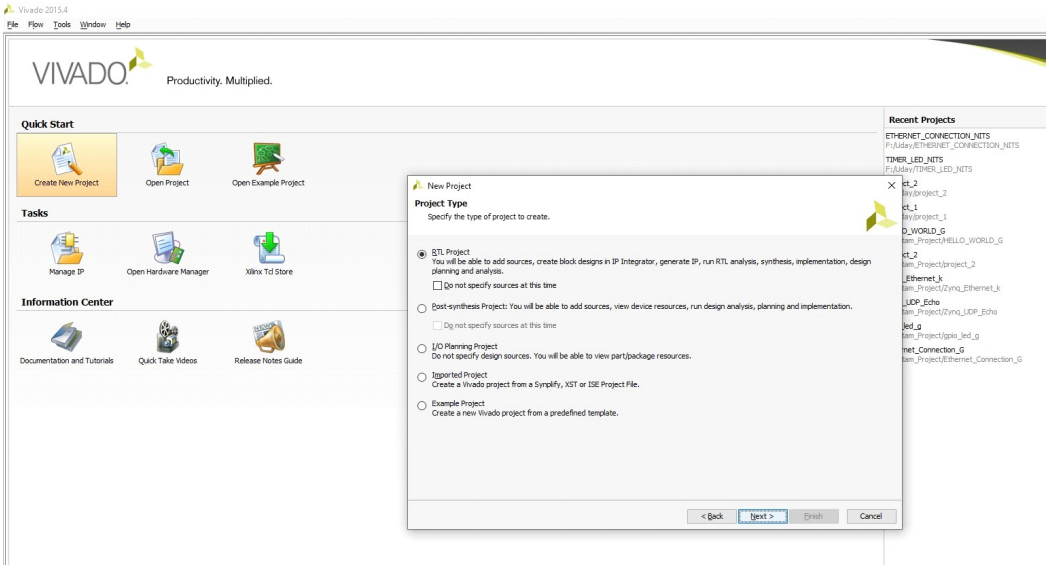
1. Create your own folder on the Desktop
2. Open the Vivado 2015.4
3. Click on new project



- Give a name to the file and select the location -> Next.

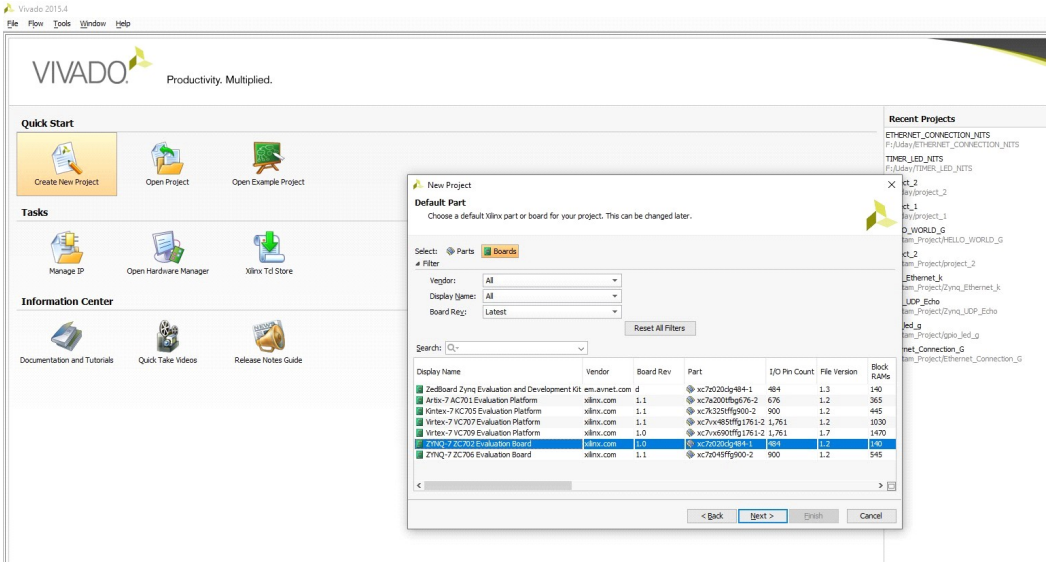


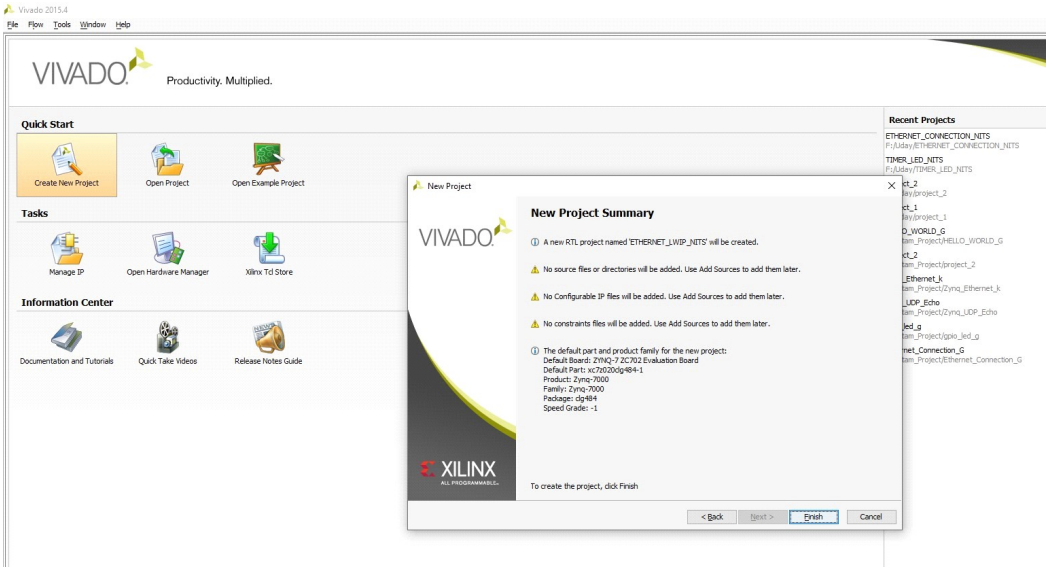
- Select Project Type -> RTL Project



- Skip Add Sources, Add Existing IP, Add Constraints(in optional)

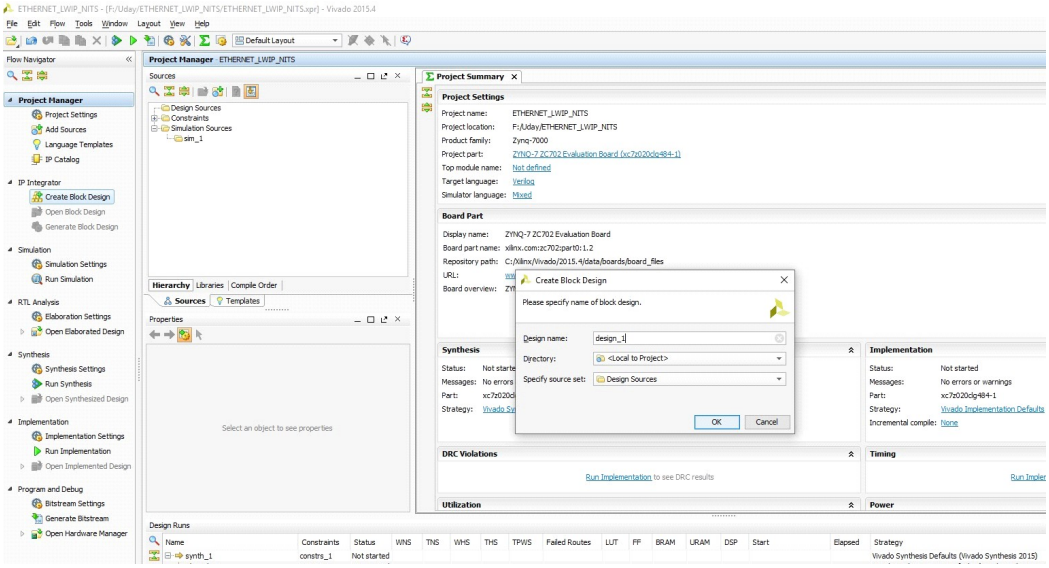
- Select Board -> ZYNQ ZC702 Evaluation Board -> Finish



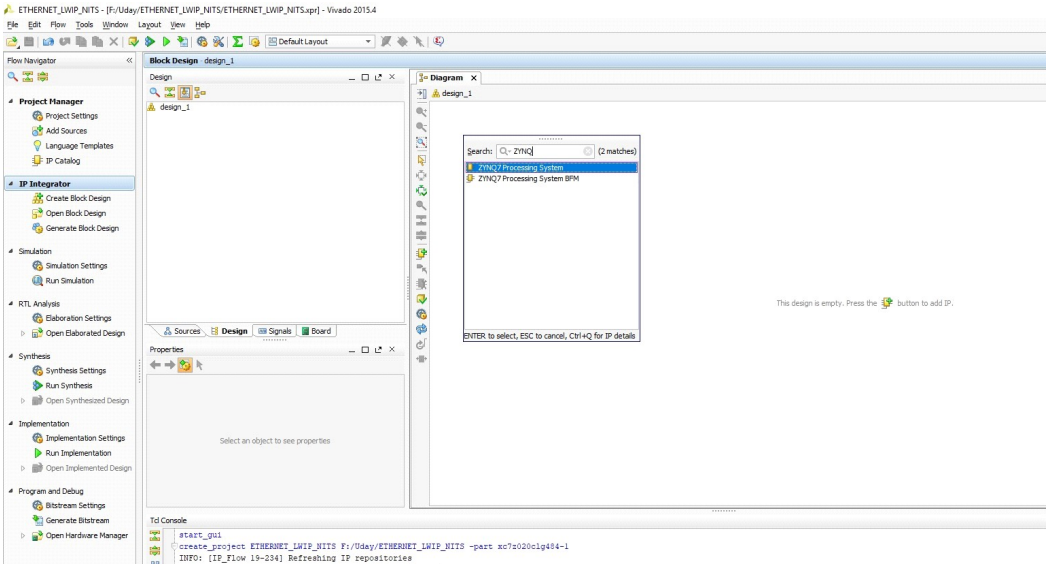


Step 2: Creating Block Diagram

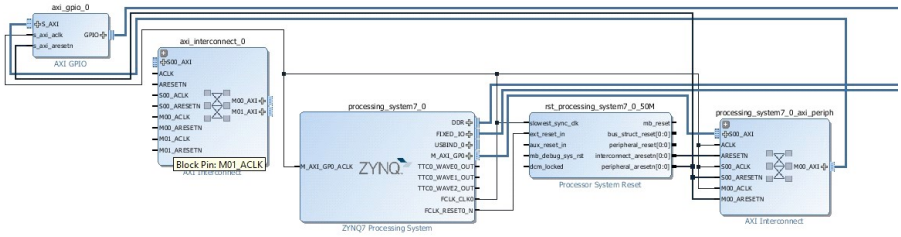
- Flow Navigator -> IP Integrator -> Create Block Design



- Add the components by using Add IP Button according to the circuit.

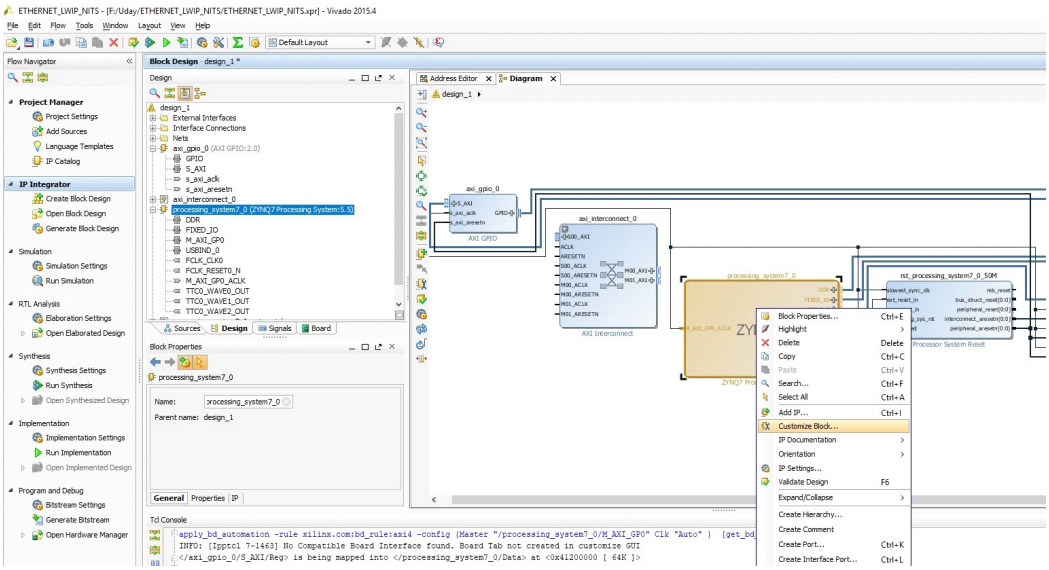


- Add ZYNQ Processing system and make the connection as per circuit diagram.

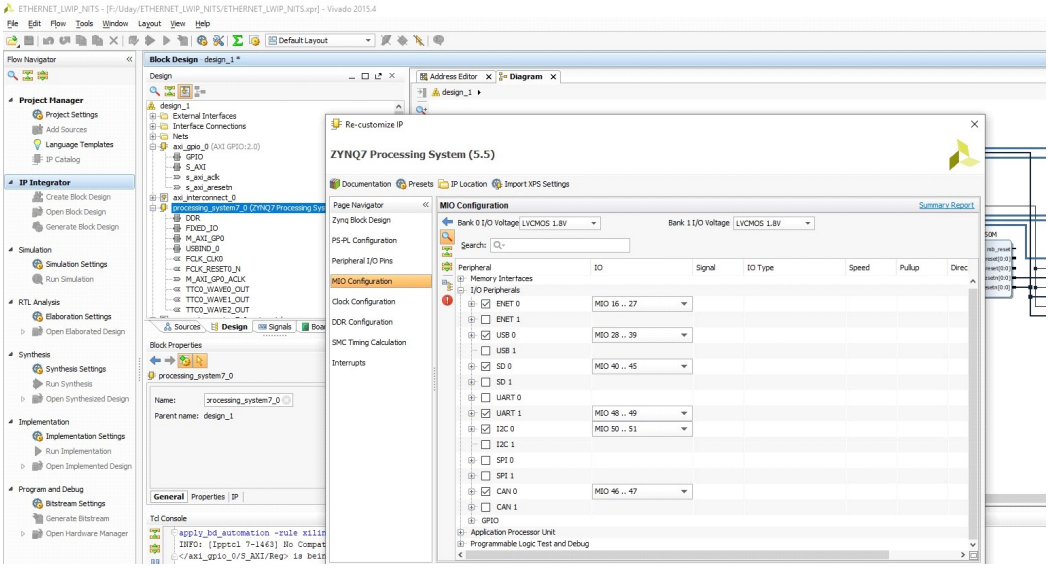


Step 3: Customizing the block.

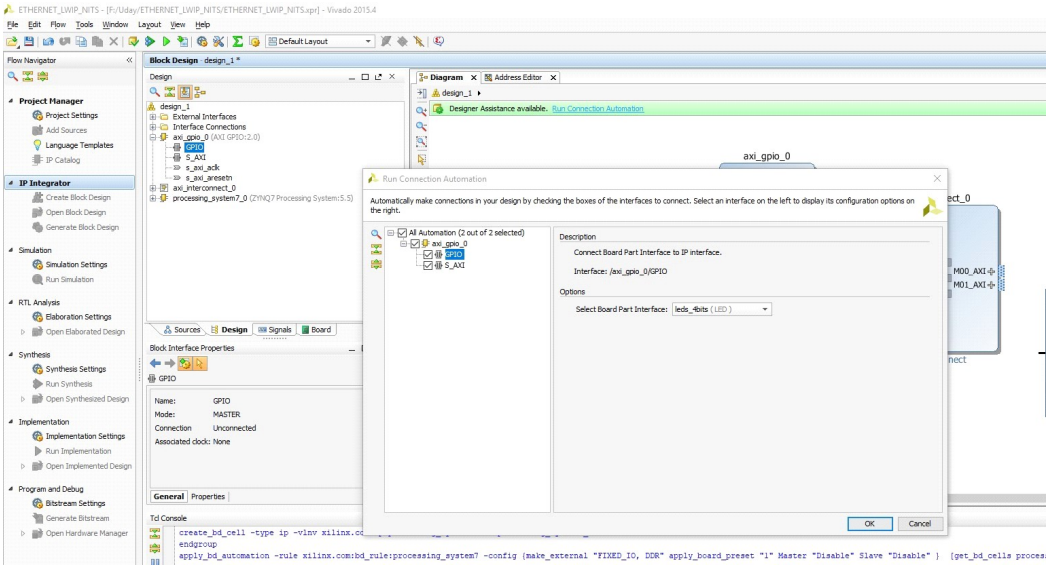
- Right click on the block -> select “customize block” option.



- Select “UART 1, USB 0, ENET 0” in I/O Properties from “MIO Configuration”.

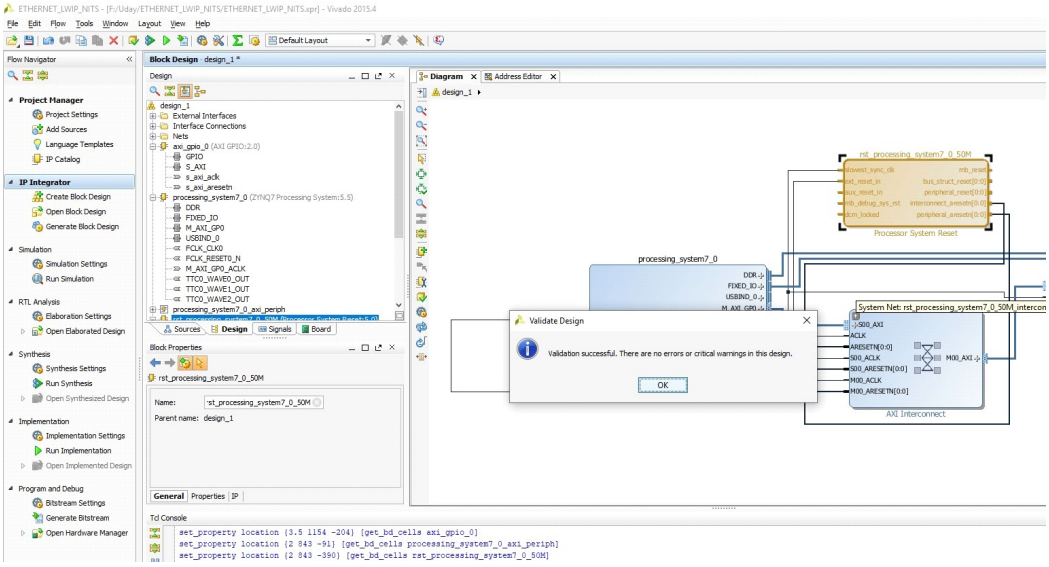


- Double click on the AXI GPIO block to customize it to led 4bits.



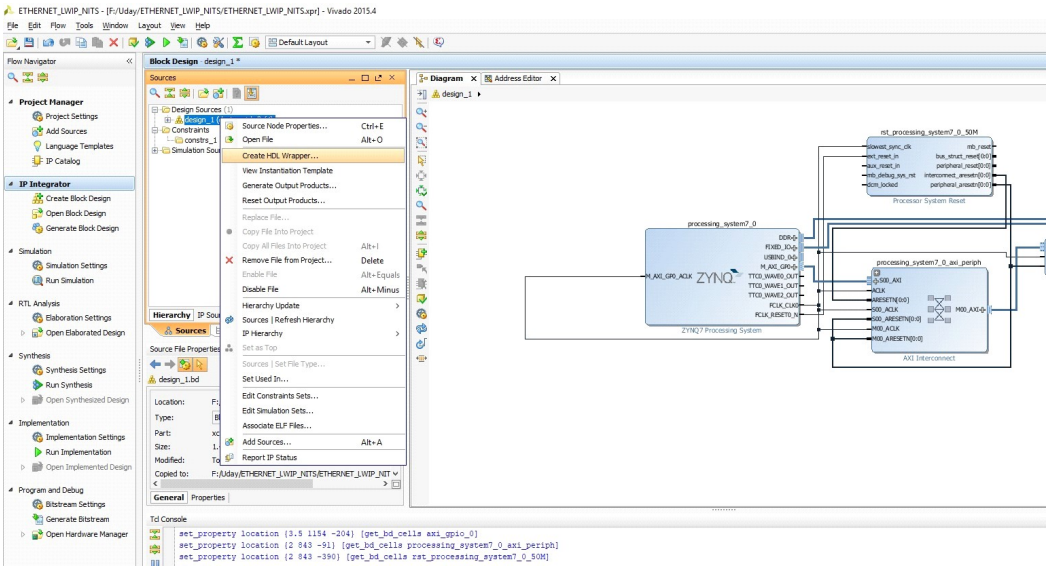
- Right click on the Processing System and select Run Block Automation -> ok
- Select the run connection automation available in designer assistance.

- Validate your design by pressing F6 key.

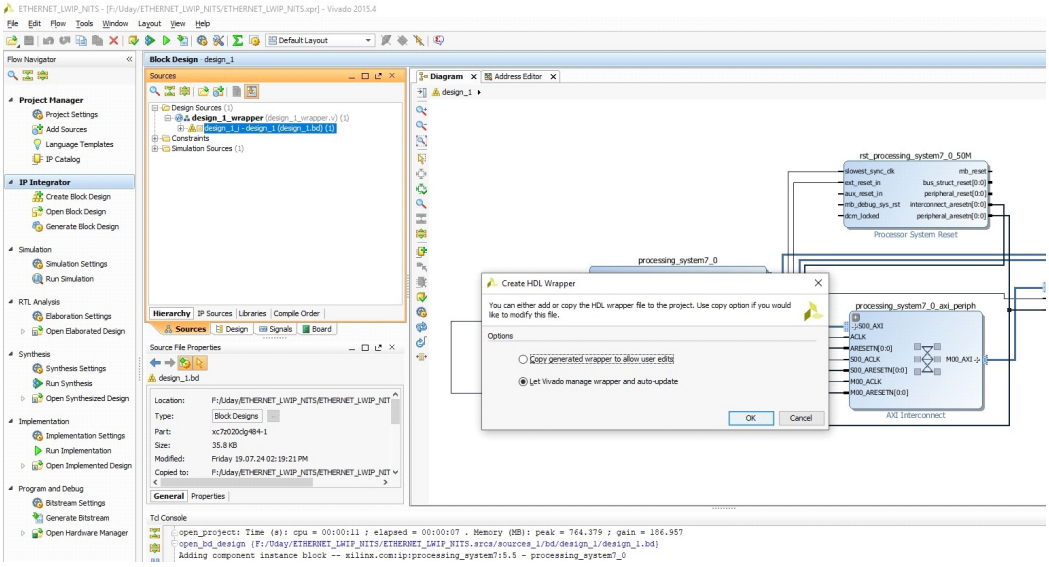


Step 4: Creating HDL wrapper.

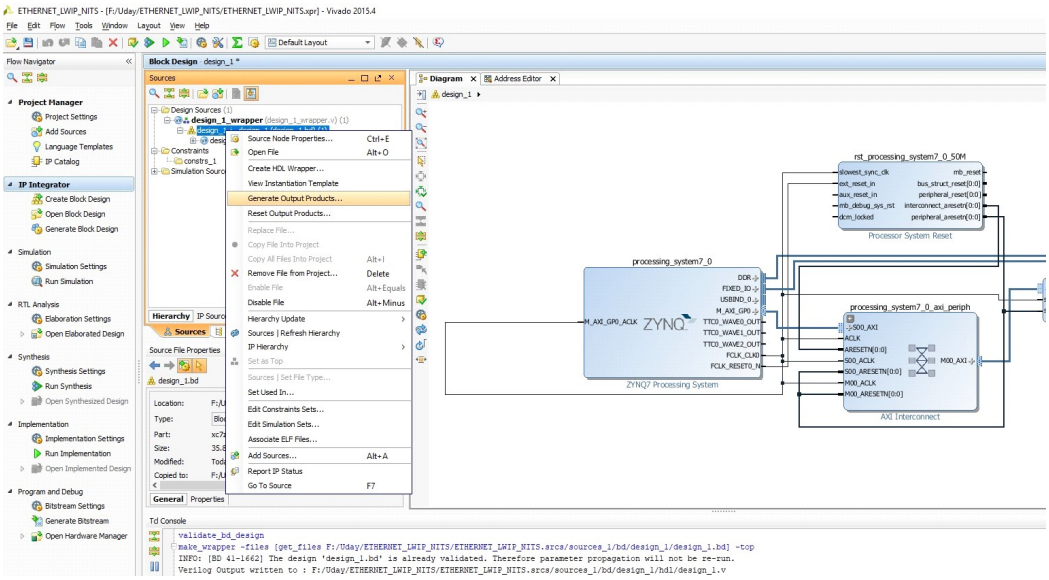
- Go to sources -> Right click on the file(Design_1) -> Create HDL wrapper



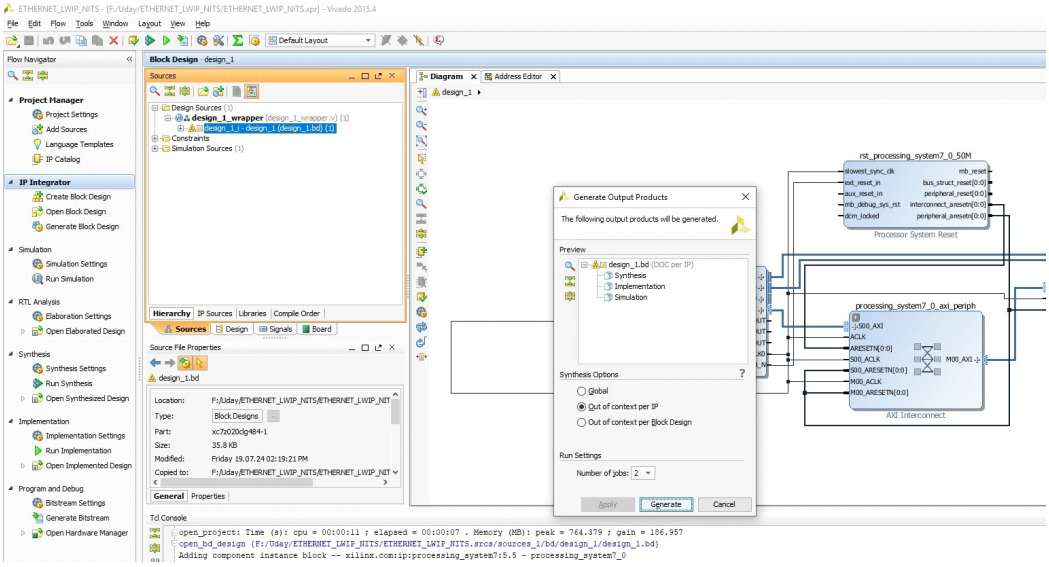
- Select “Let Vivado manage wrapper and auto-update.”



- Sources -> Right click on file -> select Generate Output Products.

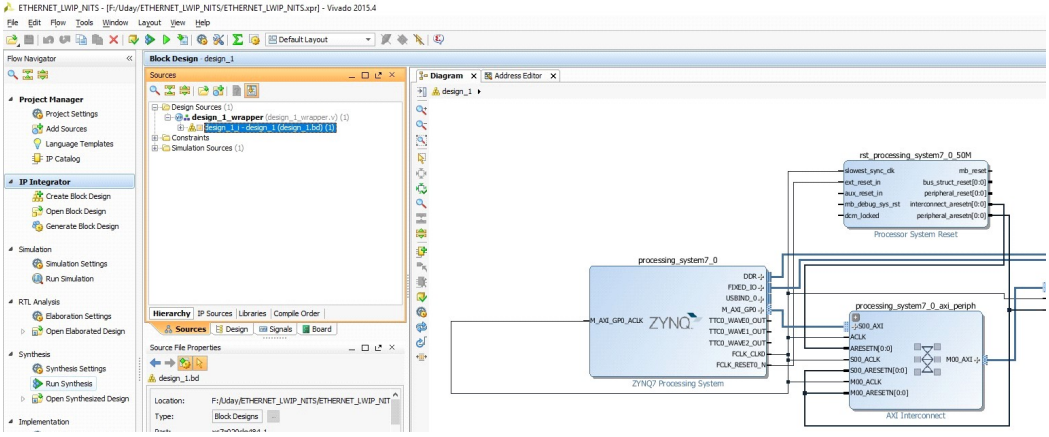


- Select -> Out of context per IP



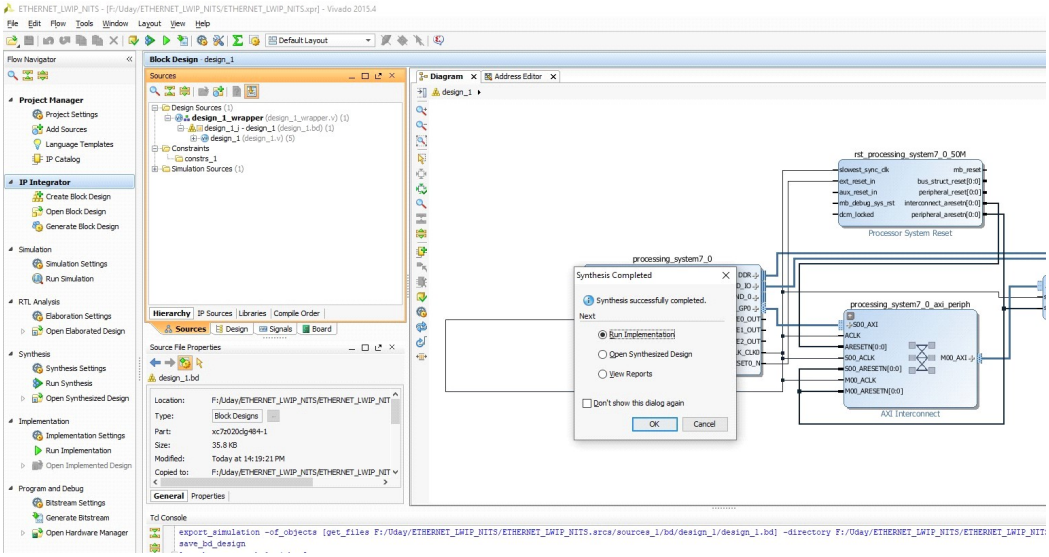
Step 5: Synthesis

- Flow Navigator -> Synthesis -> Run Synthesis



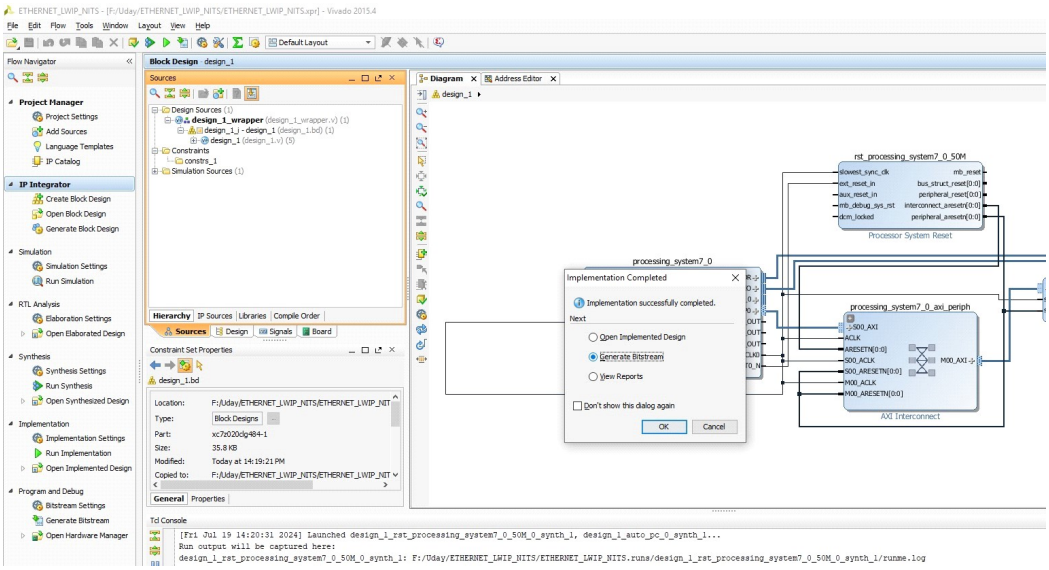
Step 6: Implementation

- Select -> Run Implementation

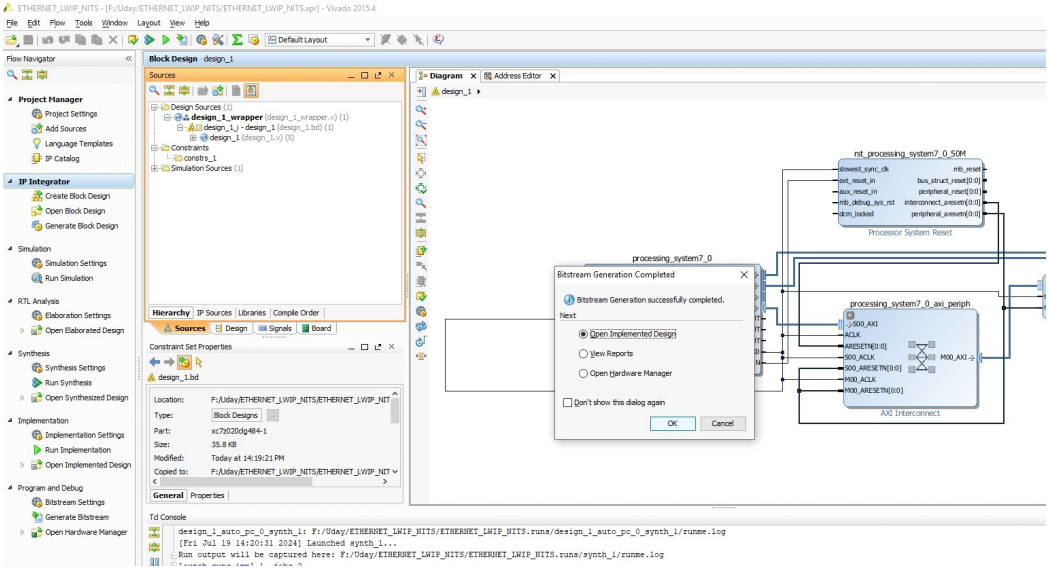


Step 7: Generating Bitstream

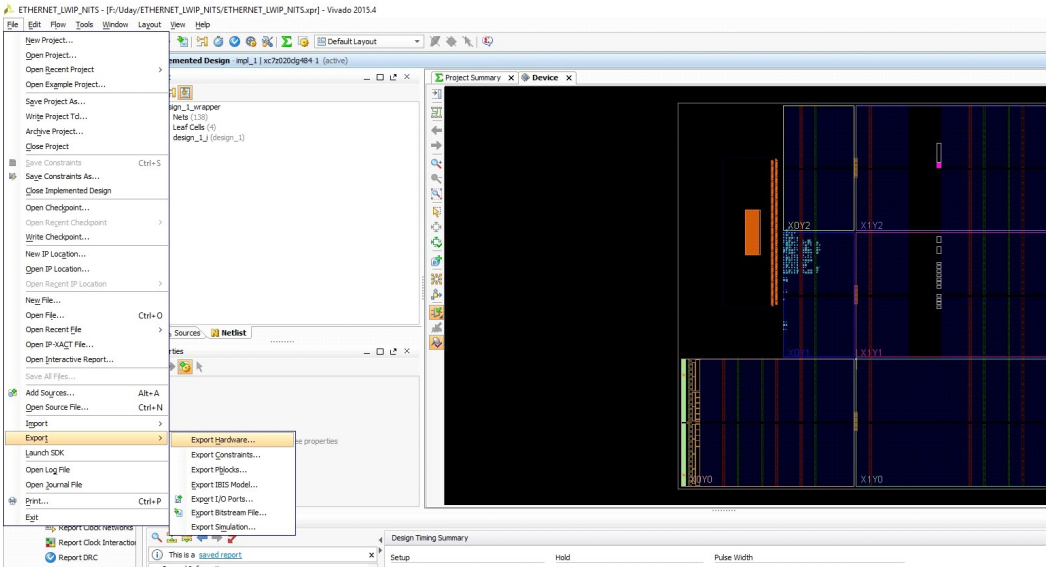
- Select Generate Bitstream -> ok.



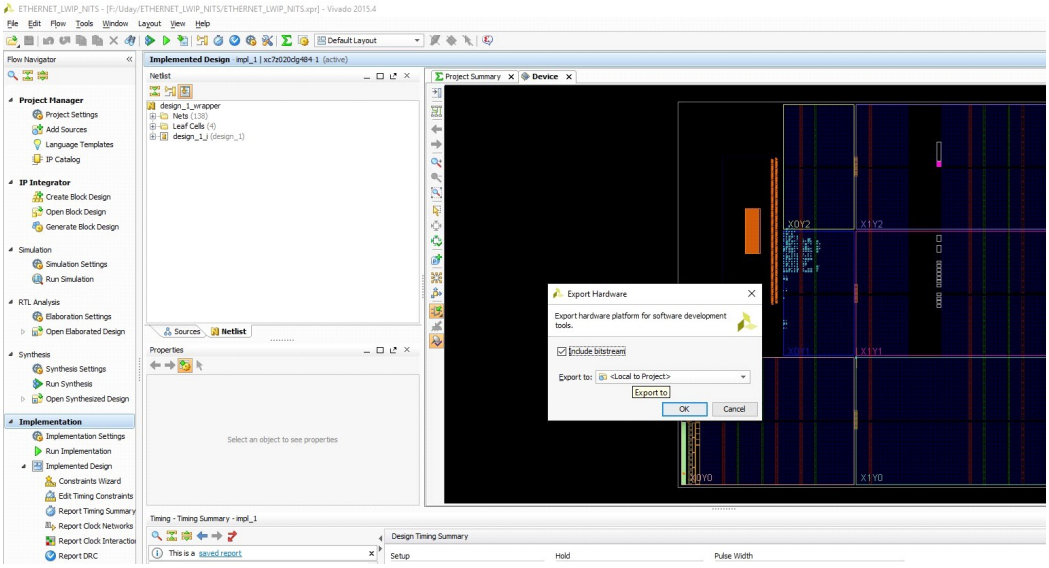
- Select “open implemented design.”



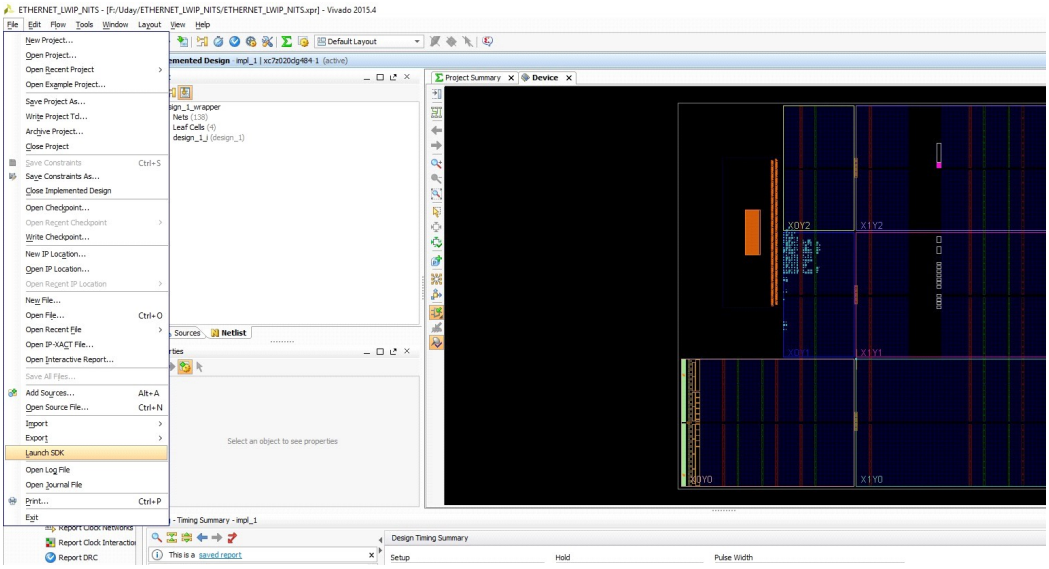
- File -> Export -> Export Hardware



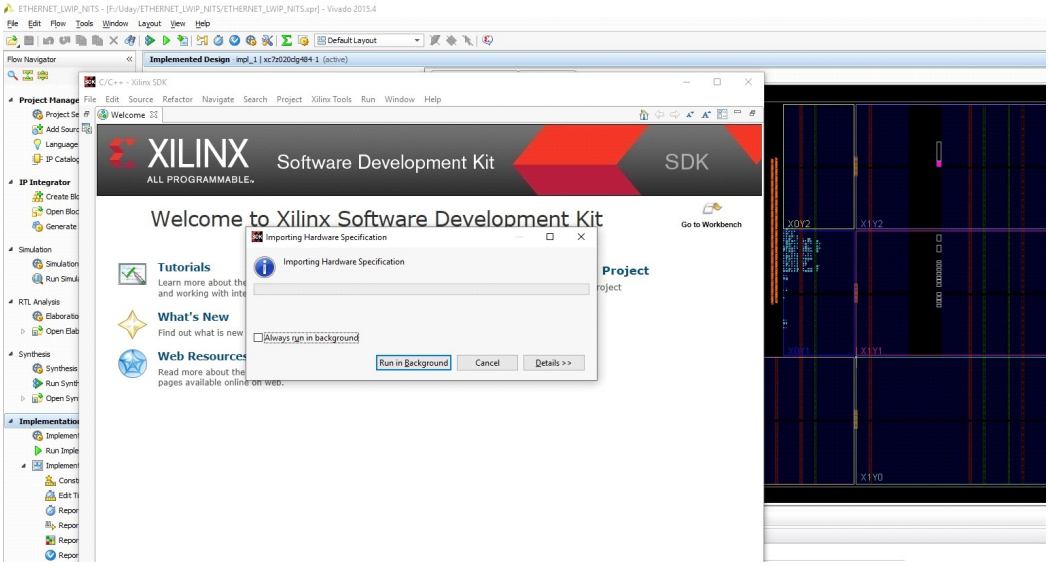
- Select “Include Bitstream.”



- File -> Launch SDK -> Ok.

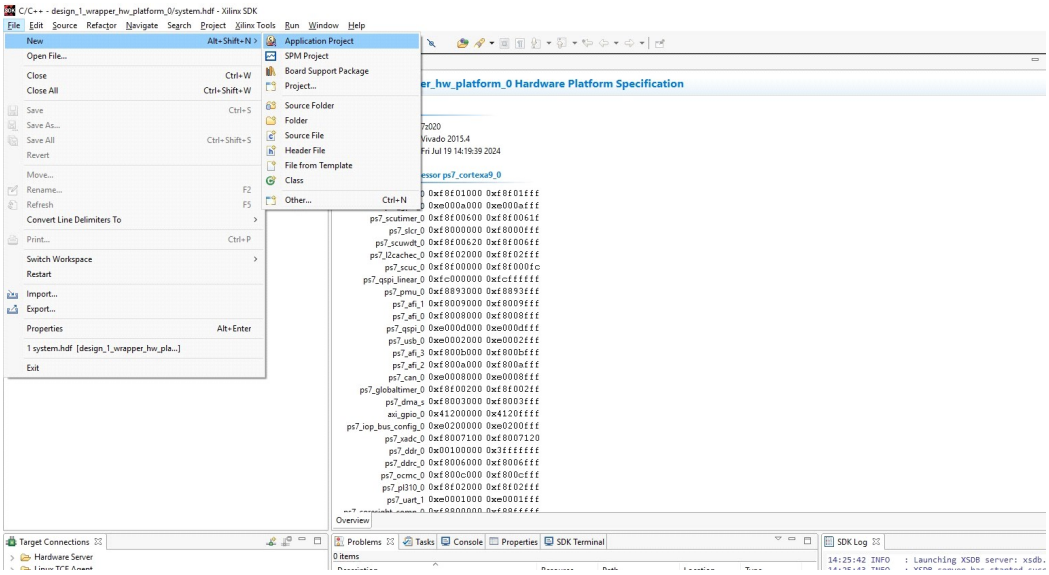


- SDK Terminal is opened.

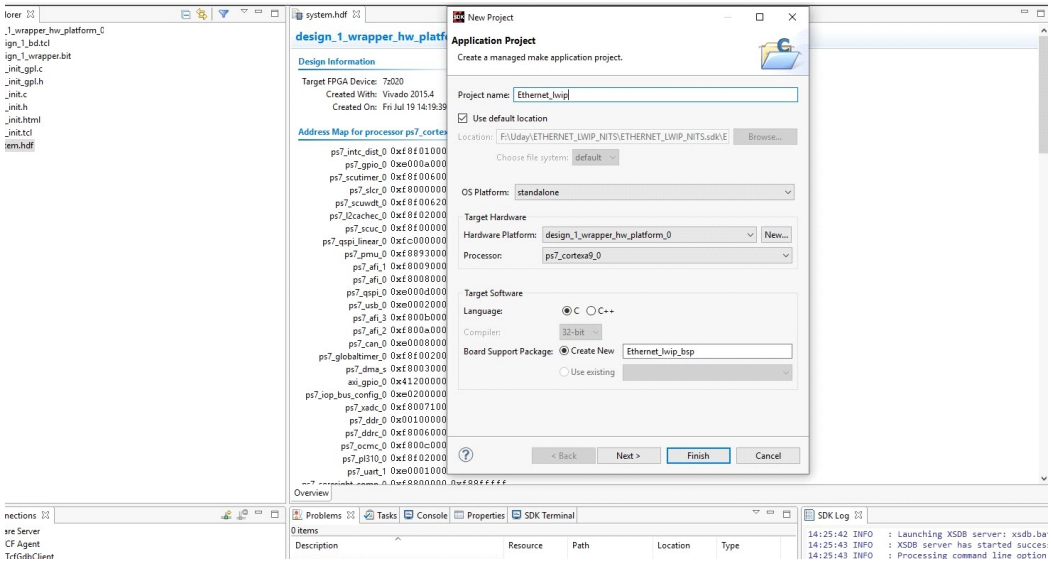


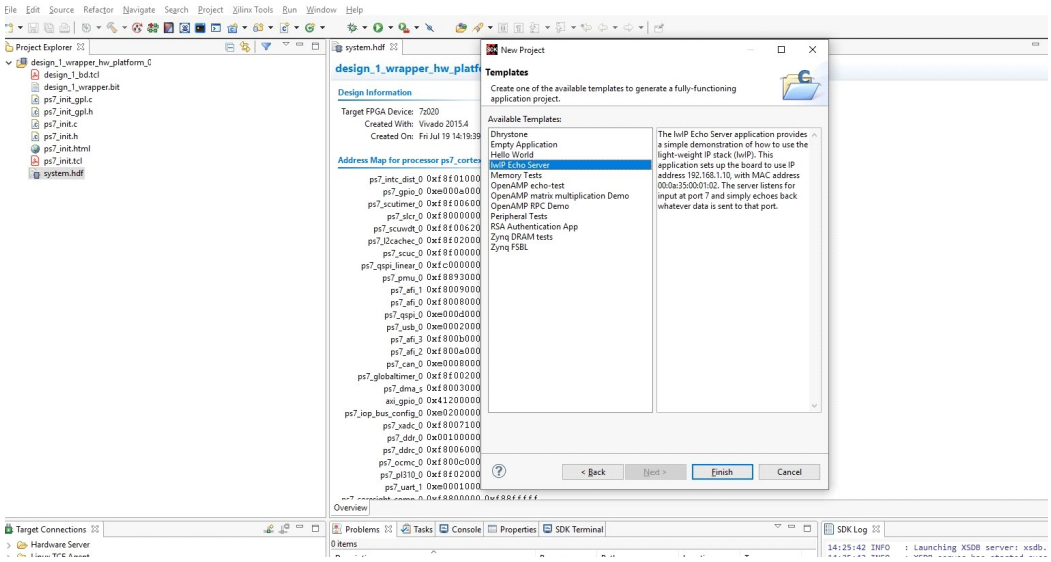
Step 8: Creating new project in SDK.

- File -> New -> Application Project



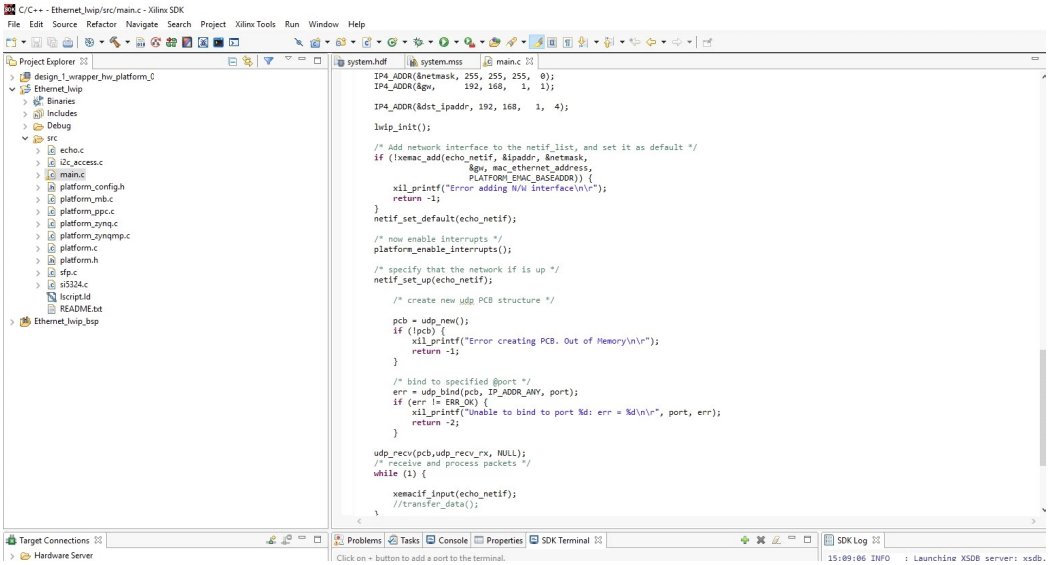
- Give File name -> next -> select “lwip echo server” and Finish.





Step 9: Open .C File.

- Right click on file name ->src -> main.c



- Type your required Code.

CODE:

```
#include <stdio.h>
#include "xparameters.h"
#include "netif/xadapter.h"
#include "platform.h"
#include "platform_config.h"
#include "xgpio.h"
#if defined (__arm__) || defined(__aarch64__)
#include "xil_printf.h"
#endif

#include "lwip/err.h"
#include "lwip/udp.h"
#include "xil_cache.h"

#define LED_CHANNEL 1

void lwip_init();
void Rest_buffer();
static struct netif server_netif;
struct netif *echo_netif;
struct ip_addr ipaddr, netmask, gw;

        struct ip_addr dst_ipaddr;
        struct udp_pcb *pcb;
        err_t err;
volatile static unsigned char Readbuffer[100];
```

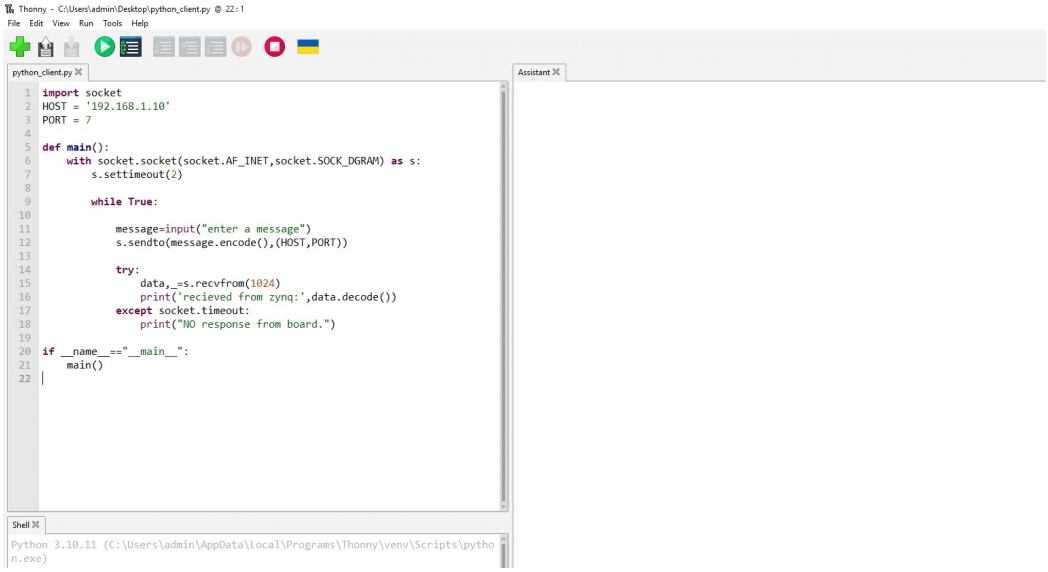
```
#if defined (__arm__) || defined(__aarch64__)
#if XPAR_GIGE_PCS_PMA_SGMII_CORE_PRESENT == 1 ||
XPAR_GIGE_PCS_PMA_1000BASEX_CORE_PRESENT == 1
int ProgramSi5324(void);
int ProgramSfpPhy(void);
#endif
#endif

char send_buf[1024]="Hello from Zynq!";

XGpio Gpio;
void udp_recv_rx(void *arg, struct udp_pcb *pcb, struct pbuf *p, struct ip_addr *addr,
u16_t port)

{
        if(p==NULL){
                        xil_printf("no data recieved\n");
                        return;
                }
                xil_printf("recieved data:%s\n",(char *)p->payload);
                u32 led_value=0;
                //int i;
                //for( i=0; i<p->len && i<4; i++)
                //{
                        led_value = ((u8 *)p->payload);

                //}
                XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, led_value);
                sleep(1);
                led_value= led_value & 0x00;

                XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, led_value);



                struct pbuf *response_pbuf= pbuf_alloc(PBUF_TRANSPORT,
strlen(send_buf),PBUF_RAM);
                 memcpy(response_pbuf -> payload,send_buf, strlen(send_buf) );
                        udp_sendto ( pcb,response_pbuf,addr,port);
                        pbuf_free(response_pbuf);
                        pbuf_free(p);


}

int main()
{

#if __aarch64__
        Xil_DCacheDisable();
```

```
#endif

	unsigned port = 7;
	/* the mac address of the board. this should be unique per board */
	unsigned char mac_ethernet_address[] =	{ 0x00, 0x0a, 0x35, 0x02, 0x97, 0x71 };

	echo_netif = &server_netif;

#if defined (__arm__) || defined(__aarch64__)
#if XPAR_GIGE_PCS_PMA_SGMII_CORE_PRESENT == 1 ||
XPAR_GIGE_PCS_PMA_1000BASEX_CORE_PRESENT == 1
	ProgramSi5324();
	ProgramSfpPhy();
#endif
#endif

	init_platform();
	/* initliaze IP addresses to be used */
	int status = XGpio_Initialize(&Gpio,XPAR_AXI_GPIO_0_DEVICE_ID);
		if(status!=XST_SUCCESS){
			xil_printf("GPIO INTIALIZATION FAILED\n");
			return -1;
		}
	XGpio_SetDataDirection(&Gpio, LED_CHANNEL, 0x0);

	IP4_ADDR(&ipaddr,  192, 168,   1, 10);
	IP4_ADDR(&netmask, 255, 255, 255,  0);
	IP4_ADDR(&gw,      192, 168,   1,  1);

	IP4_ADDR(&dst_ipaddr, 192, 168,   1,  4);

	lwip_init();

	/* Add network interface to the netif_list, and set it as default */
	if (!xemac_add(echo_netif, &ipaddr, &netmask,
						&gw, mac_ethernet_address,
						PLATFORM_EMAC_BASEADDR)) {
		xil_printf("Error adding N/W interface\n\r");
		return -1;
	}
	netif_set_default(echo_netif);

	/* now enable interrupts */
	platform_enable_interrupts();

	/* specify that the network if is up */
	netif_set_up(echo_netif);

		/* create new udp PCB structure */
```

```
		pcb = udp_new();
		if (!pcb) {
			xil_printf("Error creating PCB. Out of Memory\n\r");
			return -1;
		}

		/* bind to specified @port */
		err = udp_bind(pcb, IP_ADDR_ANY, port);
		if (err != ERR_OK) {
			xil_printf("Unable to bind to port %d: err = %d\n\r", port, err);
			return -2;
		}

	udp_recv(pcb,udp_recv_rx, NULL);
	/* receive and process packets */
	while (1) {

		xemacif_input(echo_netif);
		//transfer_data();
	}
	/* never reached */
	cleanup_platform();
	return 0;
}
void Rest_buffer()
{
	unsigned char var=0;
	for(var=0; var<100; var++)
	{
		Readbuffer[var]=0x0;
	}
}
```

Step 10: Write Client program in python script.

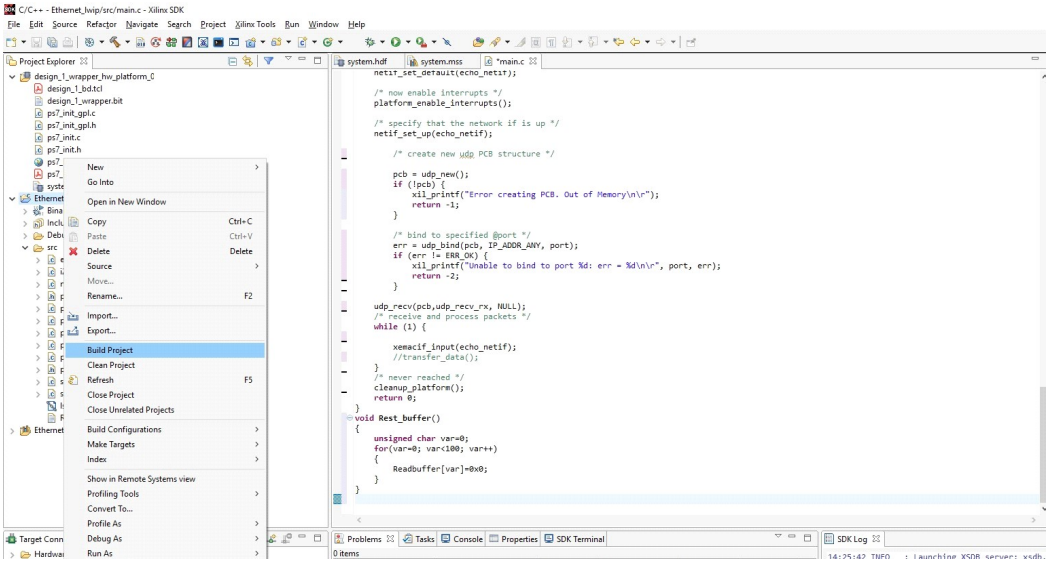
- Open Thony and write the client code and save as “Python_client.py”



CLIENT PROGRAM:

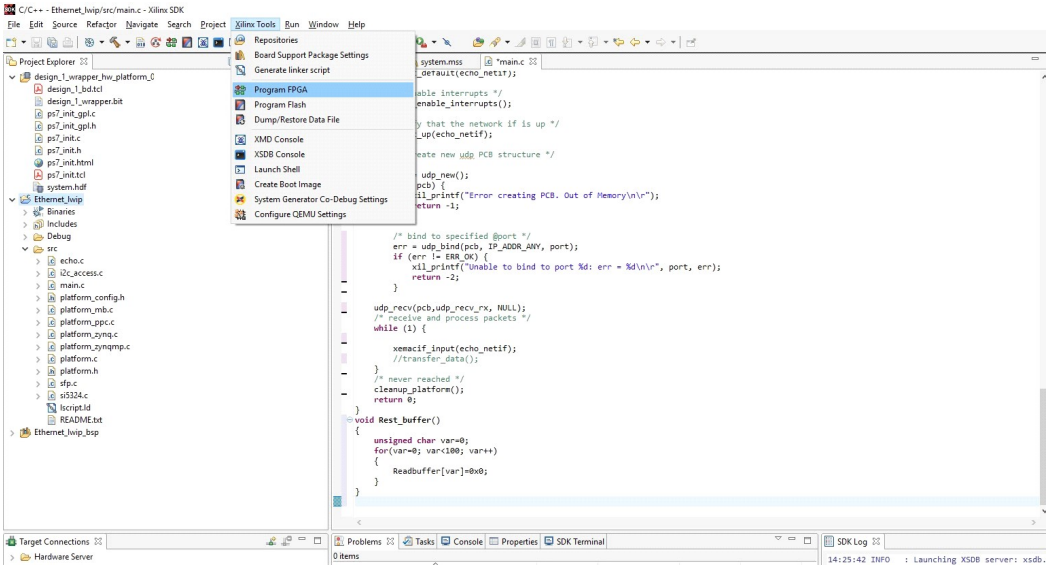
```
import socket
HOST = '192.168.1.10'
PORT = 7

def main():
   with socket.socket(socket.AF_INET,socket.SOCK_DGRAM) as s:
      s.settimeout(2)

      while True:

         message=input("enter a message")
         s.sendto(message.encode(),(HOST,PORT))

         try:
            data,_=s.recvfrom(1024)
            print('recieved from zynq:',data.decode())
         except socket.timeout:
            print("NO response from board.")

if __name__=="__main__":
   main()
```

- Right click on the file -> Build Project.

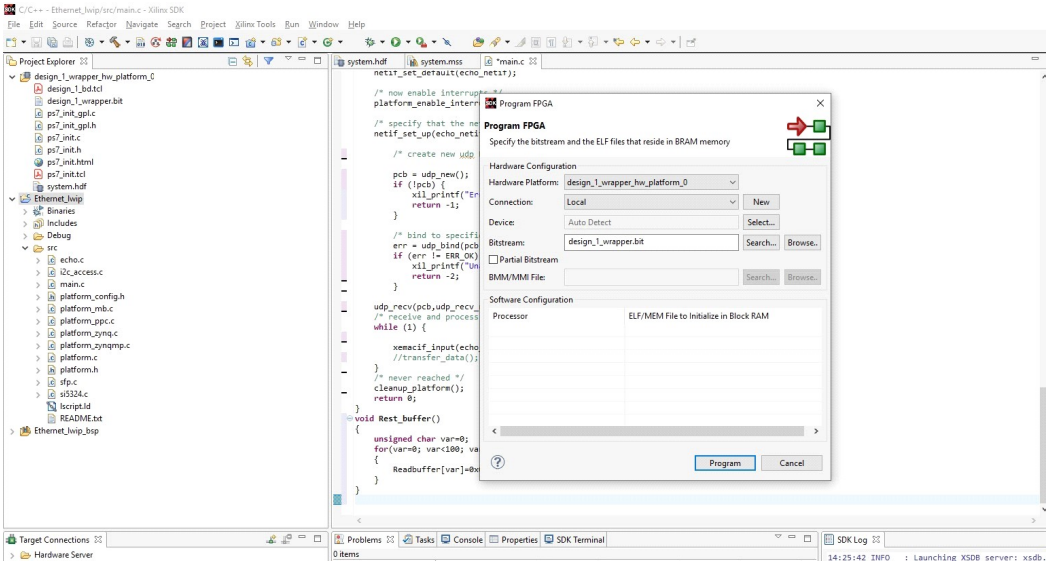


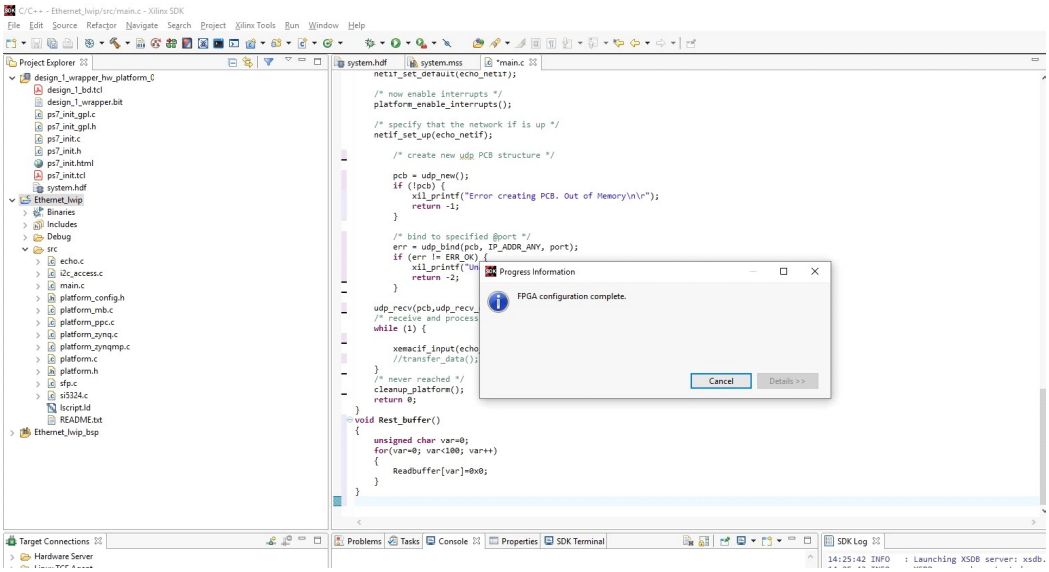
Step 11: Programming FPGA.

- Task Bar -> Xilinx Tools -> Program FPGA.



- Select Program

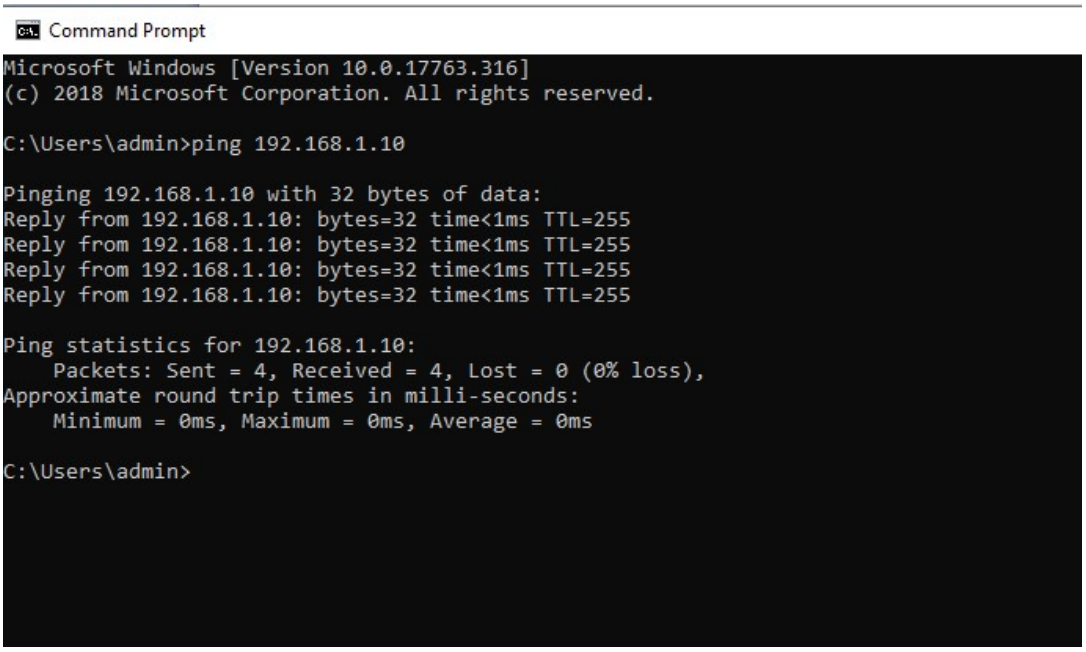




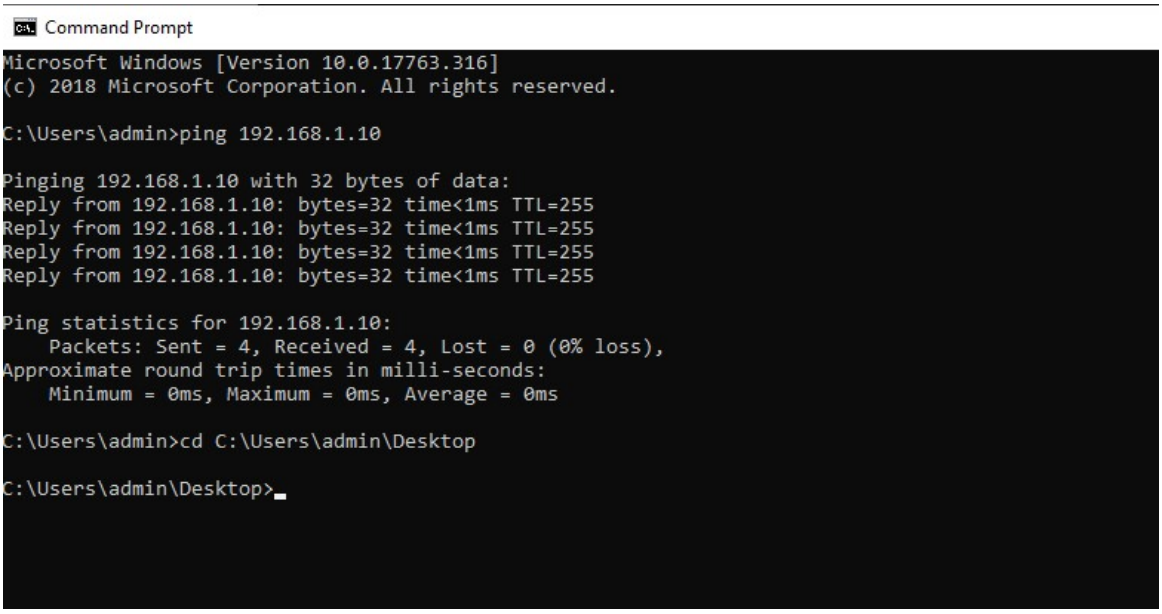
Programming FPGA is completed.

Step 11: Open command prompt.

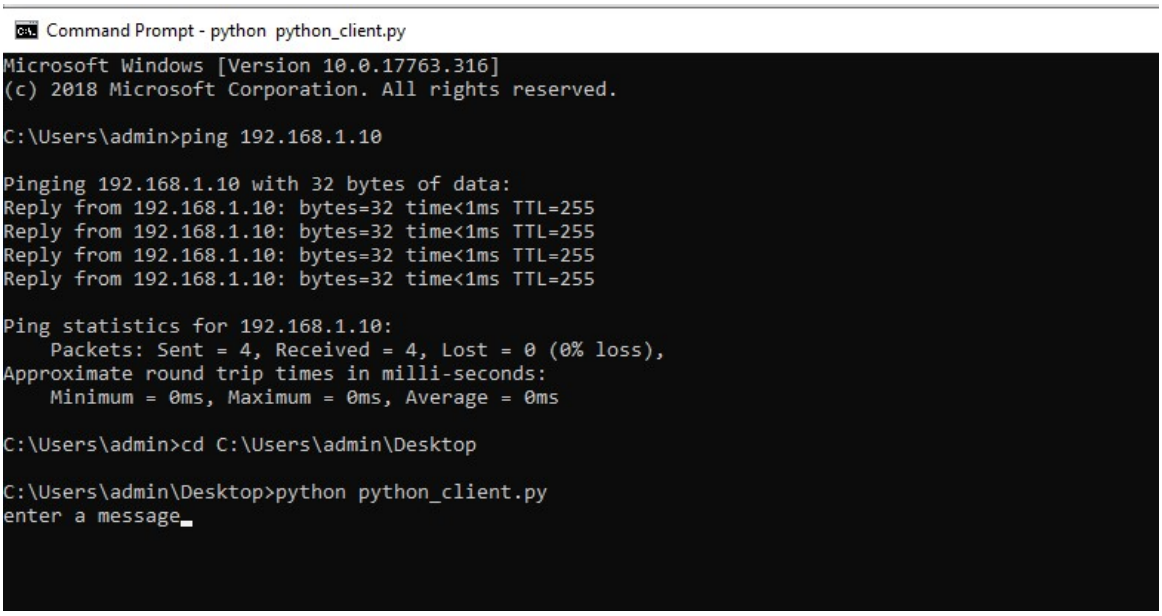
- Open command prompt and enter ping 192.168.1.10



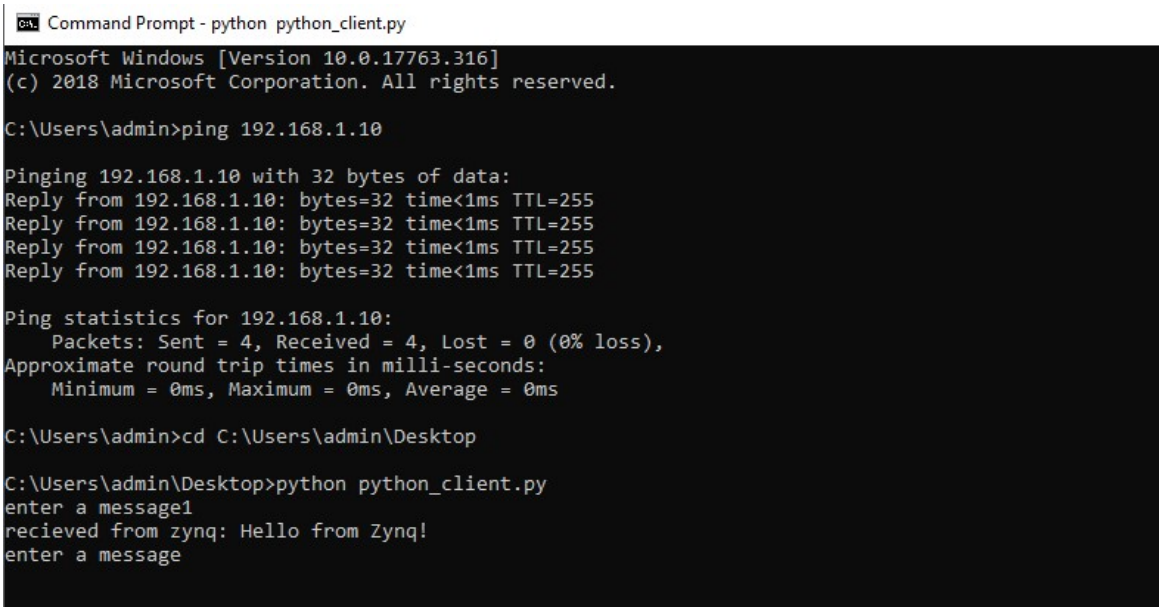
- Change the directory to the saved location of the client program .
- Type cd path\to\your\file.

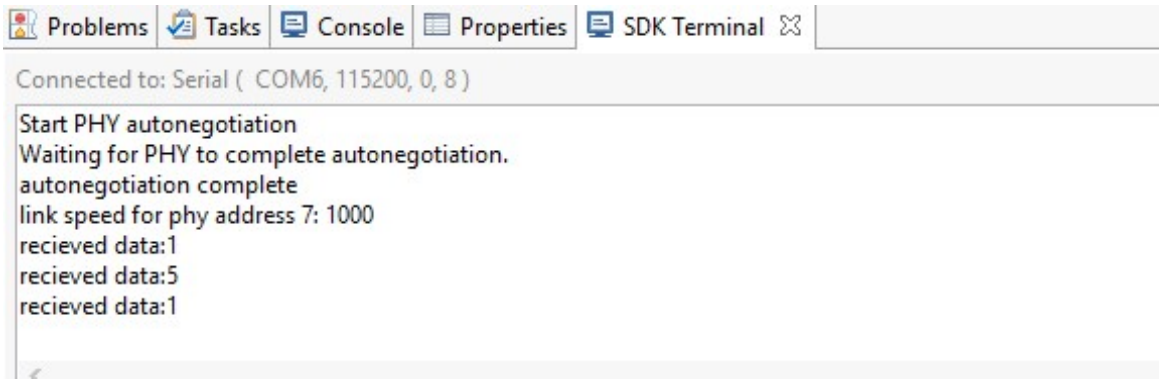


- Type python “file_name.py” and enter



- Enter any message and verify it on the serial monitor, the FPGA board and Command prompt window.
- An Acknowledgment from board is sent to pc which can be verified on the Command prompt window.





Step 12: Verifying Output

- Verify the output in command prompt, serial monitor and the FPGA board.

Result: Hence connection is established between PC and ZYNQ FPGA board and we can see the output in command prompt, serial monitor and the FPGA board.