**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **05-June-2020** | **Name:** | **Raziya Banu** |
| **Course:** | **HDL** | **USN:** | **4AL16EC058** |
| **Topic:** | **Verilog Tutorials, Building/ Demo of FPGA** | **Semester & Section:** | **8th sem & ‘B’ section** |
| **Github Repository:** |  |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report –**  In my first session today I have studied about the Verilog Tutorials, Building/ Demo of FPGA. **Understanding the FPGA: From Developing Configurations to Building a VGA Driver:** Developers love field-programmable gate arrays (FPGAs) for their reconfigurability. FPGA technology allows you to program certain functionality into the chip and then update it or even change it completely whenever you need. This technology is widely used for developing embedded systems and creating system-on-a-chip (SoC) designs.  The Apriorit team has spent a lot of time studying FPGA technology and has built numerous embedded systems with FPGAs. In this article, we talk about the origins of the FPGA, give a detailed description of an FPGA-specific framework called ISE Design Suite, and look at the main steps you need to take in order to create a VGA driver using FPGA. Understanding FPGA technology What is a [field-programmable gate array](https://en.wikipedia.org/wiki/Field-programmable_gate_array) (FPGA)? An FPGA is basically an integrated circuit that has an array of programmable logic blocks that can be reconfigured by a developer for nearly any computational purposes.  The first commercial FPGA, the XC2064, was invented by the co-founders of [Xilinx Inc.](http://www.xilinx.com/) back in 1985. Even today, Xilinx remains one of the leading manufacturers of these circuits.  An FPGA is mainly based on two technologies:   * [Programmable Logic Device](https://en.wikipedia.org/wiki/Programmable_logic_device) (PLD) – This is a programmable electronic component with a set of logic gates. Unlike a basic logic gate with a fixed function, a PLD has an undefined function when it’s manufactured, meaning you can reconfigure a PLD the way you need. Moreover, a PLD must be reconfigured before it can be used in a circuit. * [Complex Programmable Logic Device](https://en.wikipedia.org/wiki/Complex_programmable_logic_device) (CPLD) –  A CPLD is a PLD built on the idea of [personal digital assistants](https://en.wikipedia.org/wiki/Personal_digital_assistant) (PDAs), with an interconnection matrix connecting all inputs and outputs. The connection matrix is formed based on the on-chip flash memory to configure macrocells, the main building blocks of CPLDs. Macrocells contain logic that implements disjunctive normal form expressions and specific logical operations.   Over the years, developers have changed the format of FPGAs. Today, this type of integrated circuit has three programmable elements:   1. Array of logic blocks 2. Routings 3. Input/output (I/O) pads   With so many elements that a developer can reprogram for their purposes, an FPGA can perform a large number of vector processing operations simultaneously. Furthermore, an FPGA can even beat a central processing unit (CPU) in processing big data.   When talking about technical capabilities, an FPGA is also pretty similar to an [Application-Specific Integrated Circuit](https://en.wikipedia.org/wiki/Application-specific_integrated_circuit) (ASIC). Particularly, an FPGA has a high level of energy efficiency and performance, just like an ASIC. But unlike an ASIC, an FPGA can be configured for a specific objective at a much lower production cost and can be easily reconfigured after manufacturing. FPGA manufacturers As we mentioned earlier, Xilinx practically invented the FPGA as a technological solution. Today, the California-based company is one of the two biggest producers of FPGAs and the sure leader in the market.  The second biggest player on the FPGA market is [Altera](https://en.wikipedia.org/wiki/Altera), an innovative company that became part of Intel in 2015. Today, Altera and Intel produce two hybrid circuits, or system-on-a-chip (SoC) FPGAs:   * [Intel Arria 10 FPGA](https://www.intel.com/content/www/us/en/products/programmable/fpga/arria-10.html) * [Intel Stratix 10 FPGA](https://www.intel.com/content/www/us/en/products/programmable/fpga/stratix-10.html)   While Xilinx and Altera offer pretty similar solutions, some developers admit that the Altera tools have a more user-friendly and intuitive graphical user interface.  Now let’s talk about the main tasks you can solve with the help of FPGA technology. FPGA use cases For a long time, this technology was only used in telecommunications and industrial, military, aerospace, and automotive systems. However, modern FPGA integrated circuits have a much more impressive variety of applications. Thanks to their large logic gate arrays, memory blocks, and fast I/O, FPGAs can perform even the most difficult tasks.  Here are four of the most interesting field-programmable gate array application examples:   * ASIC prototyping and emulation –  Also referred to as [FPGA prototyping](https://en.wikipedia.org/wiki/FPGA_prototyping), this is a method of prototyping SoC and ASIC designs on an FPGA. Emulation and prototyping with FPGAs enables fast and accurate SoC system modeling and verification as well as accelerated software and firmware development. * Cyber security – An FPGA can be effectively used for building network intrusion detection and network intrusion prevention systems. Using an FPGA allows you to [improve network monitoring performance](https://accoladetechnology.com/public/RTC_Dont_Monitor_Your_Network_%20Without_FPGA_Acceleration.pdf#page=50). Plus, the reconfigurable nature of an FPGA makes it easier for you to update or completely change the monitoring rules. * Data center acceleration – When being used as [accelerators](https://www.intel.com/content/www/us/en/servers/accelerators/accelerators.html) for data centers, FPGAs can easily be reprogrammed for different accelerator designs. An FPGA helps you process data at high speed and increase the performance of resource-hungry processes and applications for [machine learning](https://www.apriorit.com/dev-blog/472-machine-learning-applications), [edge computing](https://www.apriorit.com/dev-blog/531-cloud-edge-computing), [data mining](https://www.apriorit.com/dev-blog/527-data-mining-cyber-security), and content delivery. * Artificial Intelligence (AI) – While there’s still a lot of debate going on among researchers, FPGAs show great potential for researching new [artificial intelligence](https://www.apriorit.com/dev-blog/474-ai-ml-cybersecurity) algorithms and training AI systems. With an FPGA’s ability to have different blocks performing different tasks, this technology can potentially provide a robust system on a chip. In fact, Microsoft was among the first companies to use an FPGA as part of an AI platform in its [Brainwave platform](https://newsroom.intel.com/news/intel-fpgas-bring-power-artificial-intelligence-microsoft-azure/) for Azure services.   In all these use cases, the ability of an FPGA to be programmed and reconfigured for multiple uses is its main benefit. So let’s see how you can configure an FPGA and what tools you can use for it. Developing an FPGA configuration Traditionally, to develop an FPGA configuration you would need to use a [Hardware Description Language](https://en.wikipedia.org/wiki/Hardware_description_language) (HDL). While there are many different HDLs, in most cases developers use one of two:   * [Verilog](https://en.wikipedia.org/wiki/Verilog) * [VHDL](https://en.wikipedia.org/wiki/VHDL)   Just as there are plenty of programming languages to choose from, FPGA producers offer numerous frameworks for developing FPGA configurations. For instance, Xilinx provides two FPGA configuration frameworks:   * [ISE Design Suite](https://www.xilinx.com/products/design-tools/ise-design-suite.html) * [Vivado Design Suite](https://www.xilinx.com/products/design-tools/vivado.html)   ISE Design Suite can be used for configuring all Virtex-6 and older FPGA families. However, you can also use this framework when working with some of the small and mid-sized Virtex-7 FPGAs.  Vivado Design Suite is a modern framework that doesn’t support older versions of FPGAs. Vivado can only be used with Virtex-7, UltraScale, and newer FPGA families.  As you can see, these frameworks support different generations of FPGA circuits released by Xilinx. The only family that’s supported by both frameworks is Virtex-7. As an alternative, you can use the [OpenCL](https://en.wikipedia.org/wiki/OpenCL" \t "_blank) framework. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **05-June-2020** | **Name:** | **Raziya Banu** | |
| **Course:** | **Udemy** | **USN:** | **4AL16EC058** | |
| **Topic:** | **Main Function in Python** | **Semester & Section:** | **8th sem & ‘B’ section** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Python main function:**In this tutorial, we will learn how to use a Python program's \_\_name\_\_ attribute to run it dynamically in different contexts.What is the main() function in Python? Some programming languages have a special function called main() which is the execution point for a program file. Python interpreter, however, runs each line serially from the top of the file and has no explicit main() function.  Python offers other conventions to define the execution point. One of them is using the main() function and the \_\_name\_\_ property of a python file. What is \_\_name\_\_ in Python? The \_\_name\_\_ variable is a special builtin Python variable that shows the name of the current module.  It has different values depending on where we execute the Python file. Let's look at an example. Running Python File as a Script Suppose we have a Python file called **helloworld.py** with the following content:  print(\_\_name\_\_)  Run Code  If we run **helloworld.py** from the command line, then it runs as a Python script. We can run the Python program using the following command:  python helloworld.py  **Run Code**  When we run the program as a script, the value of the variable \_\_name\_\_ is set to \_\_main\_\_. So the output of the following program will be:  \_\_main\_\_ Running Python file as a Module We can also run a Python file as a module. For this, we have to import this file into another Python program. Let's look at an example.  Suppose we have a Python file called **main.py** in the same directory as the heloworld.py file. It has the following content:  import helloworld  When we run this file, we will have the following output:  helloworld  Here, we can see that importing a module also runs all the code in the module file.  But, we can see that instead of displaying \_\_main\_\_, the program displays the name of the module i.e. helloworld.  It is because, in the context of running a Python file as a module, the name of the module itself is assigned to the \_\_name\_\_ variable. Using if conditional with \_\_name\_\_ Now that we have understood how \_\_name\_\_ variable is assigned values, we can use the if conditional clause to run the same Python file differently in different contexts.  Let's look at an example.  Suppose we change the content of the **helloworld.py** file to the following:  def main():  print("Hello World")  if \_\_name\_\_=="\_\_main\_\_":  main()  Run Code  Now, when we run it as a script via the command line, the output will be:  Hello World | | | |