**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **17-07-2020** | **Name:** | **Karthik J** |
| **Course:** | Introduction to FPGA Design for Embedded Systems | **USN:** | **4AL16EC030** |
| **Topic:** | Week 3 | **Semester & Section:** | **8TH A** |
| **GitHub Repository:** | Karthik-J |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| The Quartus® Prime software organizes and manages the elements of your design within a *project*. The  project encapsulates information about your design hierarchy, libraries, constraints, and project settings.  Click **File > New Project Wizard** to create a new project quickly and specify basic project settings.  When you open a project, a unified GUI displays integrated project information. The Project Navigator  allows you to view and edit the elements of your project. The Messages window lists important information  about project processing.  You can save multiple revisions of your project to experiment with settings that achieve your design goals.  Quartus Prime projects support team-based, distributed work flows and a scripting interface.  Quick Start  To quickly create a project and specify basic settings, click **File > New Project Wizard**.  New Project Wizard  Project Management Best Practices  The Quartus Prime software provides various options for setting up a project. The following best practices  help ensure efficient management and portability of your project files.  Setting and Project File Best Practices  • Avoid manually editing Quartus Prime data files, such as the Quartus Prime Project File (**.qpf**),  Quartus Prime Settings File (**.qsf**), Quartus IP File (**.qip**), or Qsys System File (**.qsys**). Typos in these  files can cause software errors. For example, the software may ignore settings and assignments.  Every Quartus Prime project revision automatically includes a supporting **.qpf** that preserves various  project settings and constraints that you enter in the GUI or add with Tcl commands. This file  contains basic information about the current software version, date, and project-wide and entity level  settings. Due to dependencies between the **.qpf** and **.qsf**, avoid manually editing **.qsf** files.  • Do not compile multiple projects into the same directory. Instead, use a separate directory for each  project.  • By default, the Quartus Prime software saves all project output files, such as Text-Format Report Files  (**.rpt**), in the project directory. Instead of manually moving project output files, change your project  compilation settings to save them in a separate directory.  To save these files into a different directory choose **Assignments** > **Settings**. Turn on the **Save project**  **output files in specified directory** option and specify a directory for the output files.  Project Archive and Source Control Best Practices  • Click **Project** > **Archive Project** to archive your project for revision control.  As you develop your design, your Quartus Prime project directory contains a variety of source and  settings files, compilation database files, output, and report files. You can archive these files using the  Archive feature and save the archive for later use or place it under revision control.  **1.** Choose **Project** > **Archive Project** > **Advanced** to open the **Advanced Archive Settings** dialog box.  **2.** Choose a file set to archive. For example, choose **File set** > **Source control with incremental**  **compilation and Rapid Recompile database** to save the source and database file required to recreate  your project with your Rapid Recompile revisions.  **3.** Add additional files by clicking **Add** (optional).  To restore your archived project, choose **Project** > **Restore Archived Project**. Restore your project  into a new, empty directory.  IP Core Best Practices  • Do not manually edit or write your own **.qsys** or **.qip** file. Use the Quartus Prime software tools to  create and edit these files.  **Note:** When generating IP cores, do not generate files into a directory that has a space in the directory  name or path.  • When you generate an IP core using the IP Catalog, the Quartus Prime software generates a **.qsys** (for  Qsys-generated IP cores) or **.qip** file. Always add the generated **.qsys** or **.qip** to your project. Do not  add the parameter editor generated file (**.v** or **.vhd**) to your design without the **.qsys** or **.qip** file.  Otherwise, you cannot use the IP upgrade or IP parameter editor feature.  **Note:** For Qsys-generated IP cores, adding the **.qsys** file to the project instead of the **.qip** file simplifies  modifying the IP with the parameter editor.  • Plan your directory structure ahead of time. Do not change the relative path between a **.qsys** file and  it's generation output directory. If you must move the **.qsys** file, ensure that the generation output  directory remains with the **.qsys** file.  • Do not add IP core files directly from the **/quartus/libraries/megafunctions** directory in your project.  Otherwise, you must update the files for each subsequent software release. Instead, use the IP Catalog  and then add the **.qip** to your project.  • Do not use IP files that the Quartus Prime software generates for RAM or FIFO blocks targeting older  device families (even though the Quartus Prime software does not issue an error).  • When generating a ROM function, save the resulting **.mif** or **.hex** file in the same folder as the  corresponding IP core's **.qsys** or **.qip** file. For example, moving all of your project's **.mif** or **.hex** files to  the same directory causes relative path problems after archiving the design.  • Always use the Quartus Prime ip-setup-simulation and ip-make-simscript utilities to generate  simulation scripts for each IP core or Qsys system in your design. These utilities produce a single  simulation script that does not require manual update for upgrades to Quartus Prime software or IP  versions. Refer to *Generating Version-Independent IP and Qsys Simulation Scripts* for details.  **Related Information**  **Generating a Combined Simulator Setup Script** on page 1-36  Viewing Basic Project Information  View basic information about your project in the Project Navigator, Report panel, and Messages window.  View project elements in the Project Navigator ( **View > Utility Windows > Project Navigator**). The  Project Navigator displays key project information, including design files, IP components, and revisions  of your project. Use the Project Navigator to:  • View and modify the design hierarchy (**right-click > Set as Top-Level Entity**)  • Set the project revision (**right-click > Set Current Revision**)  • View and update logic design files and constraint files (**right-click > Open**)  • Update IP component version information (**right-click > Upgrade IP Component**)  Viewing Project Reports  The Report panel (**Processing > Compilation Report**) displays detailed reports after project processing,  including the following:  • Analysis & Synthesis reports  • Fitter reports  • Timing analysis reports  • Power analysis reports  • Signal integrity reports  Analyze the detailed project information in these reports to determine correct implementation. Rightclick  report data to locate and edit the source in project files. Managing Project Settings  The New Project Wizard helps you initially assign basic project settings. Optimizing project settings  enables the Compiler to generate programming files that meet or exceed your specifications.  The **.qsf** stores each revision’s project settings.  Click **Assignments > Settings** to access global project settings, including:  • Project files list  • Synthesis directives and constraints  • Logic options and compiler effort levels  • Placement constraints  • Timing constraint files  • Operating temperature limits and conditions  • File generation for other EDA tools  • Target device (click **Assignments > Device**)  The Quartus Prime Default Settings File (*<revision name>***\_assignment\_defaults.qdf**) stores initial  settings and constraints for each new project revision.  QPS5V1  2016.02.09 Message Suppression Guidelines 1-7 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date:** | | **17-07-2020** | **Name:** | **Karthik J** |  |
| **Course:** | | Trail Head Sales force Developer | **USN:** | **4AL16EC030** |  |
| **Topic:** | | Platform | **Semester & Section:** | **8th A** |  |
|  | **AFTERNOON SESSION DETAILS** | | | | |
|  | **Image of session**    At Salesforce, we group our services by clouds. There’s Sales Cloud for CRM,  Service Cloud for customer support, and a handful of other clouds that help  companies support their business functions. And while each of these clouds serves  a unique purpose, there’s one thing they all have in common: the power of the  Salesforce platform.  What is the Salesforce platform, exactly?  Like any platform, the Salesforce platform is a group of technologies that supports  the development of other technologies on top of it. What makes it unique is that the  platform supports not only all the Salesforce clouds, but it also supports custom  functionality built by our customers and partners. This functionality ranges from  simple page layouts to full-scale applications.  If you’re here today, we’re assuming you know a bit about software development.  Throughout this module, we’re going to give you an overview of development on  the Salesforce platform. We talk about some of the pillars of Salesforce  development and how they work together to create a robust system. We even touch  on some common questions that developers new to the platform run into as they get  started.  Before we continue, let’s make sure we’re on the same page. If you’re brand new to  Salesforce and you haven’t completed the  Salesforce Platform Basics module  , we  suggest you do that before you keep reading.  Once you’re done with that, you’re ready to get started!  Platform Building Blocks  As we mentioned, the platform not only forms the foundation of core Salesforce  products like Sales Cloud and Service Cloud, but it also lets you build your own  functionality. Building your own functionality can mean customizing existing  Salesforce offerings or it can mean building something from scratch.  Let’s focus on that latter part and talk about what the Salesforce platform offers  developers.  Our core platform lets you develop custom data models and applications for  desktop and mobile. And with the platform behind your development, you can build  robust systems at a rapid pace.  And then there’s the Heroku platform. Heroku gives developers the power to build  highly scalable web apps and back-end services using Python, Ruby, Go, and more.  It also provides database tools to sync seamlessly with data from Salesforce.  And then there’s the host of Salesforce APIs. These let developers integrate and  connect all their enterprise data, networks, and identity information.  And then there’s the Mobile SDK. The Mobile SDK is a suite of technologies that  lets you build native, HTML5, and hybrid apps that have the same reliability and  security as the Salesforce app.  And then... wait. Let’s stop for a second.  The problem with the platform and all its parts is that listing them out takes a really  long time. And just talking about them doesn’t help you understand everything they  do. Let’s take a different approach and talk about what we can do with the platform.  Or, more precisely, what we can build with it.  The DreamHouse App  Let’s float a scenario. Throughout the rest of this module, we use this scenario to  explore the many exciting tools and technologies that the Salesforce platform  provides.  You’re a developer for DreamHouse Realty, a company that aggregates real estate  listings to better connect homebuyers and real estate agents. Your boss asks you to  build a new system to track real estate listings. Your internal employees will use it  to track and communicate about properties. Your partner real estate brokers will use  it to access information about customers. And your customers will view properties  and contact brokers for viewings.  Building an app like this one from scratch isn’t an easy thing to do. Taking on this  project in real life can involve a long, complicated list of functional requirements  and the implementation of special integrations for your company’s business data.  Working by yourself, it can take you months to get something out the door.  But before your stress builds and you melt into a puddle of existential dread,  remember: You’ve got the platform. And building complex business applications at  a breakneck pace is what the platform’s all about.  We’re going to show you a fully functional version of the DreamHouse app so you  can get a feel for how it was built. As we move through, we discuss important  Salesforce development concepts using the app to guide us.  Install the DreamHouse App  To follow along and practice the steps in this module, you need to install the  DreamHouse package in your Trailhead Playground. Follow the instructions here to  launch a playground and install the package. You also use this package and  playground when it’s time to complete the hands-on challenge.  Launch your Trailhead Playground by scrolling to the bottom of this page and  clicking  Launch  . If you see a tab in your org labeled Install a Package, great.  Follow the steps below.  If not, from the App Launcher (  ), find and select  Playground Starter  and follow  the steps. If you don’t see the Playground Starter app, copy  this package installation  link  and check out  Install a Package or App to Complete a Trailhead Challenge  on  Trailhead Help.  1.  Click the Install a Package tab.  2.  Paste  04tB00000009UeX  into the field.  3.  Click  Install | | | | |
|  |  | | | | |