1. AN 839: Design Block Reuse Tutorial for Intel® Arria® 10 FPGA Development Board..... 3
  1.1. Tutorial Overview.................................................................3
  1.2. Tutorial Software and Hardware.......................................................5
  1.3. Tutorial Files..................................................................................5
  1.4. Core Partition Reuse—Developer Tutorial.........................................6
    1.4.1. Step 1: Define a Core Partition.................................................8
    1.4.2. Step 2: Compile and Export the Core Partition.............................9
    1.4.3. Step 3: Create the Black Box File...............................................11
    1.4.4. Step 4: Copy Files to Consumer Project.......................................11
    1.4.5. Step 5: Hardware Verification (Optional)....................................12
  1.5. Core Partition Reuse—Consumer Tutorial..........................................13
    1.5.1. Step 1: Add Files and Run Synthesis.........................................14
    1.5.2. Step 2: Create a Partition for blinking_led...................................14
    1.5.3. Step 3: Compile the Design.....................................................15
    1.5.4. Step 4: Hardware Verification (Optional)....................................15
  1.6. Root Partition Reuse—Developer Tutorial.........................................16
    1.6.1. Step 1: Create a Reserved Core Partition....................................17
    1.6.2. Step 2: Define a Logic Lock Region...........................................18
    1.6.3. Step 3: Compile and Export the Root Partition.............................19
    1.6.4. Step 4: Copy Files to Consumer Project.......................................19
    1.6.5. Step 5: Hardware Verification (Optional)....................................20
  1.7. Root Partition Reuse—Consumer Tutorial..........................................21
    1.7.1. Step 1: Add the SDC and Root Partition.......................................22
    1.7.2. Step 2: Compile the Design.....................................................23
    1.7.3. Step 3: Hardware Verification (Optional)....................................23
  1.8. (Optional) Step 8: Device Programming........................................25
  1.9. AN 839: Design Block Reuse Tutorial Document Revision History............28
1. AN 839: Design Block Reuse Tutorial for Intel® Arria® 10 FPGA Development Board

This tutorial demonstrates how to reuse design blocks in Intel® Quartus® Prime Pro Edition projects. The Intel Quartus Prime Pro Edition software supports block-based design flows, also known as modular or hierarchical design flows. These flows enable preservation of design blocks (or logic that comprises a hierarchical design instance) within a project, as well as reuse of design blocks in other projects.

You can reuse design blocks with the same periphery interface, share a synthesized design block with another designer, or replicate placed and routed IP in another project. Design, implement, and verify core or periphery blocks once, and then reuse those blocks multiple times across different projects that use the same device. In design block reuse flows, you assign a hierarchical instance of logic as a design partition. You can then preserve, export, and reuse the partition according to the following reuse flows:

- **Core Partition Reuse**—allows reuse of a synthesized or final snapshot of a core logic design partition (LUTs, flip-flops, M20K memory, and DSP blocks) in another project.
- **Root Partition Reuse**—allows reuse of the synthesized or final snapshot of the root partition. The root partition includes periphery resources (including I/O, HSSIO, PCIe, PLLs), as well as any associated core resources, while reserving a region for subsequent development.

1.1. Tutorial Overview

This tutorial describes the core and root partition reuse flows for both the Developer and the Consumer roles. At a high level, the core and root partition reuse flows are similar. Both flows preserve and reuse a design partition as a .qdb file. The Developer defines, compiles, and preserves the block in the Developer project, and the Consumer reuses the block, along with their own logic, in one or more Consumer projects.

**Reusing Core Partitions**

Core partition reuse allows a Developer to create, preserve, and export a partition for reuse. The Developer exports the core partition as a .qdb, and then a Consumer can reuse that core partition in another project. The core partition can include only core resources, such as LUTs, flip-flops, M20K memory, and DSP blocks. To use this flow you assign the .qdb to an instance in the Consumer project.
Reusing Root Partitions

Root partition reuse enables you to export a synthesized or final snapshot of the device periphery and associated core logic. To export and reuse periphery elements, you export the root partition. The periphery resources include all the hardened IP in the device periphery (such as general purpose I/O, PLLs, high-speed transceivers, PCIe, and external memory interfaces), as well as associated core logic. The Developer also reserves a region for core logic development by the Consumer. The Developer defines this reserved region with a partition. The Developer defines at least the module's port connections as a black box file.

When you export the synthesized or final partition as a .qdb, the .qdb preserves the results of that compilation stage. When you subsequently reuse that partition in another project, the Compiler reuses the previous compilation results from the .qdb for that partition, thereby leveraging the previous design efforts of the Developer.
The tutorial includes the following modules:

- Core Partition Reuse—Developer Tutorial on page 6
- Core Partition Reuse—Consumer Tutorial on page 13
- Root Partition Reuse—Developer Tutorial on page 16
- Root Partition Reuse—Consumer Tutorial on page 21
- (Optional) Step 8: Device Programming on page 25

1.2. Tutorial Software and Hardware

This tutorial assumes a basic understanding of Verilog HDL design and the Intel Quartus Prime Pro Edition design flow. The steps in this tutorial correspond with use of the following Intel software and hardware.

- Linux installation of Intel Quartus Prime Pro Edition software version 18.1, with Intel Arria 10 device support.
- The Intel Arria 10GX FPGA Development Kit.

Note: You can also adapt this tutorial for Windows and other software or hardware configurations.

Related Information

- Intel Quartus Prime Pro Edition User Guide: Block-Based Design
  For general information about all block-based flows
- Intel Quartus Prime Pro Edition Foundation Online Training
  For training on Intel Quartus Prime Pro Edition basics

1.3. Tutorial Files

This tutorial includes a design example organized into directories that correspond with the flow (Core or Root partition reuse) and role (Developer or Consumer).

Figure 3. Tutorial Directory Structure
The Completed directories contain the final versions of all the files required to complete that tutorial module. You can use the files in the Completed directories to bypass tutorial steps, or skip to the final step of the tutorial module. The Script directories contain scripts to automatically run the flows that each module describes.

Follow these steps to use the design example files with this tutorial:

1. Locate and download the design reuse tutorial design files at:
2. Unzip the q191_design_block_reuse_tutorial.zip file.

The .zip includes the following subdirectories that extract to the q191_design_block_reuse_tutorial directory.

<table>
<thead>
<tr>
<th>Directory Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core_Partition_Reuse/Developer/</td>
<td>Contains the source files for completing the core partition reuse Developer tutorial steps.</td>
</tr>
<tr>
<td>Core_Partition_Reuse/Consumer/</td>
<td>Contains the source files for completing the core partition reuse Consumer tutorial steps.</td>
</tr>
<tr>
<td>Core_Partition_Reuse/Completed/</td>
<td>Contains final versions of all the files to complete the core partition reuse tutorial module.</td>
</tr>
<tr>
<td>Root_Partition_Reuse/Developer/</td>
<td>Contains the files for completing the root partition reuse Developer tutorial steps.</td>
</tr>
<tr>
<td>Root_Partition_Reuse/Consumer/</td>
<td>Contains the files for completing the root partition reuse Consumer tutorial steps.</td>
</tr>
<tr>
<td>Root_Partition_Reuse/Completed/</td>
<td>Contains final versions of all the files to complete the core partition reuse tutorial step.</td>
</tr>
<tr>
<td>*/Developer/Scripts</td>
<td>The Consumer and Developer directories contain Scripts subdirectories that include the following helpful scripts:</td>
</tr>
<tr>
<td>*/Consumer/Scripts</td>
<td>• Scripts/run.sh—automatically completes all steps for that flow.</td>
</tr>
<tr>
<td></td>
<td>• Scripts/restore.sh—restores the tutorial files for that flow to default state.</td>
</tr>
</tbody>
</table>

You must run these scripts from the project directory in the form of Scripts/run.sh. The Intel Quartus Prime Pro Edition executable must be in your system path.

1.4. Core Partition Reuse—Developer Tutorial

Follow the steps in this tutorial module to develop a core partition for export and reuse in a Consumer project.

Process Description

As an IP Developer, you define, place, route, and eventually export a block of code within a partition. You then pass the exported .qdb to a Consumer.
Figure 4. Core Partition Reuse Flow

Reuse of the core partition also requires that you provide a black box file to the Consumer. The black box file defines the ports and port interface types for synthesis in the Consumer project. The Compiler ignores any further RTL that you include in the black box file.

You add the black box file to the Consumer project as a source file. You assign the exported .qdb file to a partition in the Consumer design. The .qdb file contains all of the information from the Developer’s compilation snapshot. The Consumer can then reuse these compilation results in their project. The synthesis snapshot includes only synthesis results. The final snapshot includes synthesis, placement, and routing results.

Completed Tutorial Files

The Core_Partition_Reuse/Completed/Developer/ tutorial directory contains the completed files for this tutorial module.

Tutorial Steps

This tutorial module includes the following steps:

- **Step 1: Define a Core Partition** on page 8
- **Step 2: Compile and Export the Core Partition** on page 9
- **Step 3: Create the Black Box File** on page 11
- **Step 4: Copy Files to Consumer Project** on page 11
- **Step 5: Hardware Verification (Optional)** on page 12
Command-Line Alternative Step

You can skip Step 1: Define a Core Partition on page 8 and Step 2: Compile and Export the Core Partition on page 9 in this tutorial module by adding the following line to the Intel Quartus Prime Settings file (.qsf), and then running the Core_Partition_Reuse/Developer/Script/run.sh script.

```
set_instance_assignment -name PARTITION blinking_led -to \\
u_blinking_led -entity top
```

1.4.1. Step 1: Define a Core Partition

Follow these steps to open the top.qpf tutorial project in the Intel Quartus Prime Pro Edition software, run design synthesis, and define a design partition for core logic.

1. In the Intel Quartus Prime Pro Edition software, click File ➤ Open Project and open the /Core_Partition_Reuse/Developer/top.qpf project file.

2. To synthesize the design, click Analysis & Synthesis on the Compilation Dashboard. The Compilation dashboard displays a check mark when synthesis is complete.

Figure 5. Compilation Dashboard

3. In the Project Navigator, right-click u_blinking_led in the Hierarchy tab, point to Design Partition, and select the Default partition Type. A design partition icon appears next to each instance you assign.
4. To view and edit all design partitions in the project, click **Assignments ➤ Design Partitions Window**. You can also define new partitions in this window, or specify automatic export of a partition following compilation.

**Figure 7. Design Partitions Window**

1.4.2. Step 2: Compile and Export the Core Partition

Follow this step to run full compilation and manually export a final snapshot of your core partition. Alternatively, you can automate partition export each time you run compilation, as **Step 1: Create a Reserved Core Partition** on page 17 describes later in this tutorial.

1. To run full compilation of the design and create the final snapshot, click **Compile Design** on the Compilation Dashboard. Check marks indicate when each stage of compilation is complete.
2. To export the core partition, click **Project ➤ Export Design Partition**. Select `blinking_led` for the **Partition name**, and the **final** compilation **Snapshot** for export.

**Figure 8. Full Compilation Complete**

![Compilation Dashboard]

**Figure 9. Export Design Partition Dialog Box**

![Export Design Partition Dialog Box]

3. Confirm blinking_led.qdb as the Partition Database File name, and then click OK. The final blinking_led.qdb that you export preserves the complete placement and routing information from the original project when you reuse the block in a Consumer project.

Related Information

1.4.3. Step 3: Create the Black Box File

Integrating a core partition .qdb into another project requires that you add a supporting black box file to the Consumer project. The black box file defines the ports and port interface types for the block that you reuse in the Consumer project. Follow these steps to create a black box port definitions file for the blinking_led partition.

1. To create a new text file, click File ➤ New, select SystemVerilog HDL File under Design Files, and then click OK. Create a file that contains only the port definitions for the partition you export. These are the same port definitions that the blinking_led.sv file specifies:

```module blinking_led (  
    output [3:0] value,  
    input      clock  
);  
endmodule```

2. Save the black box file as blinking_led_bb.sv. When saving this file, disable the option to Add file to current project.

Figure 10. Disable Add File to Current Project

1.4.4. Step 4: Copy Files to Consumer Project

After exporting the core partition and creating the black box file, you copy the files to the Consumer project directory for subsequent use in the Core Partition Reuse—Consumer Tutorial on page 13.

Manually copy the blinking_led.qdb and blinking_led_bb.sv files to the Core_Partition_Reuse/Consumer/ directory.
Note: There is no requirement to pass an .sdc file for the partition to the Consumer project, because the .sdc does not influence the logic, placement, or routing of a finalized partition in the Consumer project. However, the Developer can optionally provide an .sdc file for the partition to the Consumer project for timing analysis of the partition after complete integration with the Consumer design.

In the Core Partition Reuse—Consumer Tutorial on page 13, you integrate the blinking_led.qdb and blinking_led_bb.sv files into the Consumer project.

1.4.5. Step 5: Hardware Verification (Optional)

You can now optionally verify the results of the Core Partition Reuse—Developer tutorial module in hardware by completing (Optional) Step 8: Device Programming on page 25.

After completing this tutorial module, LEDs D6-D3 map to the blinking_led core, and LEDs D10-D7 map to the top-level design. After you configure the FPGA with the SRAM Object File (.sof), the blinking_led core flashes red LEDs in a binary counting order. The top-level design does not illuminate any LEDs.

Figure 11. Illumination of LEDs After Core Partition Reuse—Developer Tutorial Module

Off

Counting

D10-D7

D6-D3
1.5. Core Partition Reuse—Consumer Tutorial

Follow the steps in this tutorial module to reuse a core partition in a Consumer project.

**Process Description**

As a core partition Consumer, you receive the final core partition that the Developer provides. The Consumer adds the black box file and assigns the `.qdb` in the Consumer project. Because the exported `.qdb` includes compiled netlist information, the Consumer project must target the same FPGA device part number, and use the same Intel Quartus Prime version as the Developer project.

**Figure 12. File Exchange Between Developer and Consumer**

The Consumer must specify all signals and port directions, as well as any Verilog HDL parameters or VHDL generics. The Developer can optionally include an `.sdc` for the partition to verify the partition timing results after full integration in the Consumer project.

**Completed Tutorial Files**

The Core_Partition_Reuse/Completed/Consumer/ tutorial directory contains the completed files for this tutorial module.

**Tutorial Module Steps**

This tutorial module includes the following steps:

- **Step 1: Add Files and Run Synthesis** on page 14
- **Step 2: Create a Partition for blinking_led** on page 14
- **Step 3: Compile the Design** on page 15
- **Step 4: Hardware Verification (Optional)** on page 15
Command-Line Alternative Step

You can skip Step 1: Add Files and Run Synthesis on page 14 through Step 3: Compile the Design on page 15 in this tutorial module by adding following lines to the .qsf, and then running the Core_Partition_Reuse/Consumer/Script/run.sh script:

```bash
#Create the partition
set_instance_assignment -name PARTITION blinking_led -to u_blinking_led -entity top
#Assign the .qdb file to the partition
set_instance_assignment -name QDB_FILE_PARTITION blinking_led.qdb -to u_blinking_led -entity top
#Add the BB file to the file list
set_global_assignment -name SYSTEMVERILOG_FILE blinking_led_bb.sv
```

1.5.1. Step 1: Add Files and Run Synthesis

To add the core partition to the Consumer project, add the black box as a source file in the project, and assign the core partition .qdb to an instance in the Design Partitions Window, as the following steps describe:

1. In the Intel Quartus Prime Pro Edition software, click File ➤ Open Project and open the /Core_Partition_Reuse/Consumer/top.qpf project file.
2. To add files to the project, click Project ➤ Add/Remove Files in Project.
3. On the Files pane, click the browse (...) button near File name to locate and select the /Core_Partition_Reuse/Consumer/blinking_led_bb.sv black box file. Click Open, and then click OK. The file is now a source file in the project.

![Figure 13. Adding Black Box File to the Project](image)

4. To synthesize the design, click Analysis & Synthesis on the Compilation Dashboard. The Compilation dashboard displays a check mark when synthesis is complete.

1.5.2. Step 2: Create a Partition for blinkingLed

Follow these steps to create a partition for blinkingLed.

1. In the Project Navigator, right-click u_blinkingLed in the Hierarchy tab, point to Design Partition, and select the Default partition Type.
2. To assign the blinkingLed.qdb to the partition, click Assignments ➤ Design Partitions Window.
3. Double-click in the Partition Database File cell, and then click browse (...) to select the blinkingLed.qdb from the Developer project.
1.5.3. Step 3: Compile the Design

After creating a partition for `blinking_led`, you are ready to run a full compilation of the design.

1. To run a full compilation, click **Compile Design** on the Compilation Dashboard.
2. View the results of compilation in the Compilation Report.

After you complete these steps, the project uses the `blinking_led.qdb` file as a source, in place of the RTL. The placement and routing from the previous partition export is preserved. The logic in the top-level design is synthesized, placed, and routed, while preserving the placed and routed `blinking_led.qdb` partition.

1.5.4. Step 4: Hardware Verification (Optional)

You can now optionally verify the results of the Core Partition Reuse—Consumer Tutorial module in hardware by completing **(Optional) Step 8: Device Programming** on page 25.

After completing this tutorial module, LEDs D6-D3 map to the `blinking_led` core, and LEDs D10-D7 map to the top-level design. After configuring the FPGA, the `blinking_led` core flashes red LEDs in a binary counting order. The top-level design shows a shifting bit in green.

Figure 15. **Illumination of LEDs After Core Partition Reuse—Consumer Tutorial Module**

<table>
<thead>
<tr>
<th>D10-D7</th>
<th>D6-D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shifting Bit</td>
<td>Counting</td>
</tr>
</tbody>
</table>
1.6. Root Partition Reuse—Developer Tutorial

Follow the steps in this tutorial module to develop a root partition for reuse.

**Process Description**

As a root partition Developer, you create, compile, and eventually export a top-level root partition that includes the device periphery. The Developer also reserves a core partition for subsequent development of core logic in the Consumer project.

**Figure 16. Root Partition Reuse Flow**

![Root Partition Reuse Flow Diagram](image)

The Developer fully compiles this design and then provides the root partition .qdb and an .sdc file for the top-level design.

The following figure shows the Developer's root partition with a placeholder for the `blinking_led` module. For this tutorial, `blinking_led` simply ties the outputs to 4'b1111 to turn off the LEDs.

**Figure 17. RTL View of Developer's Root Partition**
Completed Tutorial Files

The Root_Partition_Reuse/Completed/Developer/ tutorial directory contains the completed files for this tutorial module.

Tutorial Module Steps

This tutorial module includes the following steps:

- **Step 1: Create a Reserved Core Partition** on page 17
- **Step 2: Define a Logic Lock Region** on page 18
- **Step 3: Compile and Export the Root Partition** on page 19
- **Step 4: Copy Files to Consumer Project** on page 11
- **Step 5: Hardware Verification (Optional)** on page 20

Command-Line Alternative Tutorial Step

You can skip Step 1: Create a Reserved Core Partition on page 17 through Step 3: Compile and Export the Root Partition on page 19 in this tutorial module by adding the following lines to the .qsf, and then running the Root_Partition_Reuse/Developer/Script/run.sh script.

```
set_instance_assignment -name PARTITION blinking_led -to u_blinking_led -entity top
set_instance_assignment -name PERIPHERY_REUSE_CORE ON -to u_blinking_led -entity top
set_instance_assignment -name PLACE_REGION "X63 Y102 X185 Y162" -to u_blinking_led
set_instance_assignment -name RESERVE_PLACE_REGION ON -to u_blinking_led
set_instance_assignment -name CORE_ONLY_PLACE_REGION ON -to u_blinking_led
set_instance_assignment -name ROUTE_REGION "X63 Y102 X185 Y162" -to u_blinking_led
set_instance_assignment -name EXPORT_PARTITION_SNAPSHOT_FINAL root_partition.qdb -to u_blinking_led -entity top
```

1.6.1. Step 1: Create a Reserved Core Partition

To export and reuse the root partition, first create a reserved core partition for subsequent core logic development in the Consumer project. When you define the partition, you can optionally specify settings to automatically export the partition each time you run compilation.
1. In the Intel Quartus Prime Pro Edition software, click **File ➤ Open Project** and open the `/Root_Particle_Reuse/Developer/top.qpf` project file.

2. To synthesize the design, click **Analysis & Synthesis** on the Compilation Dashboard.

3. In the Project Navigator, right-click `u_blinking_led` in the **Hierarchy** tab, point to **Design Partition**, and select the **Reserved Core** partition type.

4. To automatically export the final root_partition each time you run the Fitter, click **Assignments ➤ Design Partitions Window**.

5. Specify a `.qdb` file name for the **Post Final Export File** option for the partition. The `.qdb` exports to the directory you specify, or to the project directory by default.

**Figure 18. Setting Reserved Core Partition and Post Final Export File**

1.6.2. Step 2: Define a Logic Lock Region

You must define a core-only, reserved, fixed routing region to reserve core resources in the Consumer project for the reserved core partition. The Consumer uses this area for core logic development.

Ensure that the exclusive placement region size is large enough to contain all core logic in the reserved core partition. For projects with multiple core partitions, constrain each partition in a non-overlapping placement or routing region. Follow these steps to define a core-only, reserved, fixed routing region to reserve core resources in the Developer project for non-periphery development:

1. Right-click the `u_blinking_led` instance in the **Project Navigator** and click **Logic Lock Region ➤ Create New Logic Lock Region**.

2. To modify the region properties, click **Assignments ➤ Logic Lock Regions Window**.

3. In the **Origin** column, specify `X63_Y102`.

4. Change the **Width** to **123**, and the **Height** to **61**.

5. Enable the **Reserved** and **Core-Only** options.

6. Click the **Routing Region** cell. The **Logic Lock Routing Region Settings** dialog box appears.

7. Specify **Fixed with expansion** with **Expansion Length** of **0** for the **Routing Type**. The actual size and location are arbitrary for this tutorial. However, you can view and adjust the Logic Lock Region shape in the Chip Planner.
8. In the Logic Lock Regions window, right-click the \texttt{u\_blinking\_led} Logic Lock region name, and then click \textbf{Locate Node ➤ Locate in Chip Planner}.

The Logic Lock region is shaded in purple. Preserving the periphery requires exporting everything outside the Logic Lock region. This is the reverse of the core partition reuse flow.

### 1.6.3. Step 3: Compile and Export the Root Partition

After compilation, the root partition exports automatically at the final stage, according to your specifications for the \textbf{Post Final Export File} option in the Design Partitions Window.

Follow these steps to compile the design and export the root partition:

1. To run full compilation, click \textbf{Compile Design} on the Compilation Dashboard.
2. View the exported \texttt{root\_partition.qdb} in the project directory.

\textit{Note:} The .qdb exports to the project directory in your project, regardless of the directory you specify in the Design Partitions Window.

### 1.6.4. Step 4: Copy Files to Consumer Project

Manually copy the \texttt{root\_partition.qdb} and \texttt{top.sdc} files to the \texttt{Root\_Partition\_Reuse/Consumer/} directory.
Note: The top-level design requires .sdc constraints. The Consumer can also include a separate .sdc file to constrain the logic that they provide. The Logic Lock boundary is visible in the Chip Planner in the Consumer project for reference only. The Consumer cannot modify this region.

In the Root Partition Reuse—Consumer Tutorial on page 21, you integrate the root_partition.qdb and top.sdc files into the Consumer project.

1.6.5. Step 5: Hardware Verification (Optional)

You can now verify the results of the Root Partition Reuse—Developer Tutorial module in hardware by completing the steps in (Optional) Step 8: Device Programming on page 25.

After completing this tutorial module, LEDs D6-D3 map to the blinking_led core, and LEDs D10-D7 map to the top-level (root) design. When you create and load the .sof, the blinking_led core does not illuminate any LEDs. The top-level design shows a shifting bit in green.

The behavior of the periphery LED driver carries into the Consumer project via the final .qdb file.

Figure 20. Illumination of LEDs after the Root Partition Reuse—Developer Tutorial
1.7. Root Partition Reuse—Consumer Tutorial

Follow the steps in this tutorial module to reuse a root partition in a Consumer project.

**Process Description**

As a root partition Consumer, you receive the final top-level, placed, and routed root partition from the Developer. The Developer reserves a core partition for subsequent development by the Consumer. The Consumer completes the final design by adding the root partition and integrating with their own logic.

**Figure 21. Root Partition Reuse File Exchange**

The following figure shows the RTL view of the Consumer's *blinking_led* partition. The Developer compiles and exports the final snapshot as a *.qdb* file. The Consumer adds their own logic to the reserved core partition.

**Figure 22. Consumer's RTL View of *blinking_led* Partition**

Finally, the Consumer integrates their logic with the files from the Developer to create the final image.

**Figure 23. RTL View of the Integrated Developer and Consumer Partitions**
Since the root partition .qdb includes final placement and routing information for the partition, the placement and routing is identical in the Consumer project, with the exception of clock signals that may be global signals.

**Completed Files**

The Root_Partition_Reuse/Completed/Consumer/ tutorial directory contains the completed files for this tutorial module.

**Tutorial Module Steps**

This tutorial module includes the following steps:

- **Step 1: Add the SDC and Root Partition** on page 22
- **Step 2: Compile the Design** on page 23
- **Step 3: Hardware Verification (Optional)** on page 23

**Command-Line Alternative Tutorial Step**

You can skip **Step 1: Add the SDC and Root Partition** on page 22 through **Step 2: Compile the Design** on page 23 in this tutorial module by adding the following lines to the .qsf, and then running the Root_Partition_Reuse/Consumer/Script/run.sh script.

```
set_global_assignment -name SDC_FILE top.sdc
set_instance_assignment -name QDB_FILE_PARTITION root_partition.qdb -to | -entity top
```

### 1.7.1. Step 1: Add the SDC and Root Partition

Follow these steps to add the .sdc file and the root_partition.qdb to the Consumer project.

1. In the Intel Quartus Prime Pro Edition software, click **File ➤ Open Project** and open the /Root_Partition_Reuse/Consumer/top.qpf project file.
2. To add the .sdc file to the project, click **Project ➤ Add/Remove Files in Project**.
3. On the Files pane, click the browse (...) button near File name to locate and select the top.sdc file. Click **Open**, and then click **OK**.
4. In the Design Partitions Window, specify a .qdb in **Partition Database File** to replace the root_partition logic.

![Figure 24. Specifying the Partition Database File](image)
1.7.2. Step 2: Compile the Design

After adding the .sdc and root partition to the Consumer project, run a full compilation of the design.

1. To run full compilation, click Compile Design on the Compilation Dashboard.

Figure 25. Compilation Complete

2. View the results of compilation in the Compilation Report.

1.7.3. Step 3: Hardware Verification (Optional)

You can now verify the results of the Root Partition Reuse—Consumer Tutorial module in hardware by completing the steps in (Optional) Step 8: Device Programming on page 25.

After completing this tutorial module, LEDs D6-D3 map to the blinking_led core, and LEDs D10-D7 map to the top-level design. The blinking_led core flashes red LEDs in a binary counting order. The top-level design shows a single bit shifting in green.
Figure 26. Illumination of LEDs after the Root Partition Reuse—Consumer Tutorial

D10-D7  |  D6-D3

Shifting Bit  |  Counting
1.8. (Optional) Step 8: Device Programming

You can optionally configure the FPGA on the Intel Arria 10 GX Development Kit to verify the results in hardware. You can adapt the following steps if you are using a different device or development kit. Configuring the FPGA involves opening the Intel Quartus Prime Pro Edition Programmer, connecting to the Development Kit board, and loading the configuration SRAM Object File (.sof) into the SRAM of the FPGA.

Note: A .sof file configures the SRAM of an Intel FPGA. A Programmer Object File (.pof) programs a flash memory device with an FPGA configuration image for subsequent loading to an FPGA.

Follow these steps to configure the FPGA on the Intel Arria 10 GX Development Kit:

1. To open the Intel Quartus Prime Programmer, click Tools ➤ Programmer.
2. Connect the board cables:
   - JTAG USB cable to board
   - Power cable attached to board and power source
3. Turn on power to the board.
4. In the Intel Quartus Prime Programmer, click Hardware Setup.

   **Figure 27. Hardware Setup**

   ![Hardware Setup](image)

   5. In the Hardware list, select USB-BlasterII, and then click Close. The device chain appears.

      Note: If the device chain does not appear, verify the board connections.

   6. Click Auto-Detect. The device chain populates.

   7. In the Found Devices list, select the device that matches your design and click OK. For this tutorial, select the 10AX115S2 device that matches the 10AX115S2F45I1SG FPGA on the Intel Arria 10 GX Development Kit.
8. Right-click the 10AX115S2 row in the file list, and then click Change File.

9. Browse to select the top.sof file from the appropriate tutorial/output_files/directory.

10. Enable the Program/Configure option for the 10AX115S2 row.
11. Click **Start**. The progress bar reaches 100% when device configuration is complete. The device is now fully configured and in operation.

**Note:** If device configuration fails, make sure the device you select for configuration matches the device you specify during `.sof` file generation.
### 1.9. AN 839: Design Block Reuse Tutorial Document Revision History

This document has the following revision history:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.07.15</td>
<td>19.2.0</td>
<td>• Updated description of partition type GUI and column display in Design Partitions Window.</td>
</tr>
<tr>
<td>2019.04.11</td>
<td>18.1.0</td>
<td>• Updated default location of .qdb export to the project directory.</td>
</tr>
</tbody>
</table>
| 2018.11.20        | 18.1.0                      | • Removed reference to Placed snapshot. Only Synthesized and Final snapshots are supported for design block reuse.  
• Replaced references to "periphery reuse core" with "reserved core partition." This terminology has changed.  
• Described automated partition export.  
• Described Partition Database File option in Design Partitions Window.  
• Updated link to latest design example files. |
| 2018.01.15        | 17.1.0                      | • First public release. |