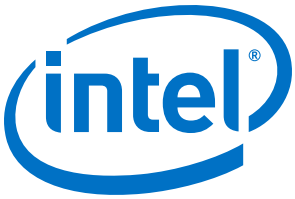


AN 824: Intel® FPGA SDK for OpenCL™ Board Support Package Floorplan Optimization Guide

AN-824
2017.08.08


Subscribe

Send Feedback

Contents

Intel® FPGA SDK for OpenCL™ Board Support Package Floorplan Optimization Guide

The *Intel® FPGA SDK for OpenCL™ Board Support Package (BSP) Floorplan Optimization Guide* provides floorplanning guidelines for OpenCL(1) BSP. It also provides guidance on how you can acquire the base seed with best average maximum operating frequency and evaluate BSP resource utilization efficiency.

This document assumes that you are familiar with OpenCL(2) concepts as described in the OpenCL Specification version 1.0 by the Khronos Group.

OpenCL BSP Compilation Flow

OpenCL BSP supports the following types of compile flows:

- **Flat compile** [`--bsp-flow flat`]: Performs a flat compilation of the entire design (BSP along with kernel generated hardware).
- **Base compile** [`--bsp-flow base`]: Performs a base compilation by using LogicLock restrictions from `base.qsf` file. The kernel clock target is relaxed so that the BSP hardware has more freedom to meet timing. A `base.qar` database is created to preserve the BSP hardware, which is the static region.
- **Import compile** [<`default`>]: Restores the timing closed static region from the `base.qar` database and compiles only the kernel generated hardware. It also increases the kernel clock target to obtain the best kernel maximum operating frequency (f_{max}).

OpenCL BSP Floorplan Partition

OpenCL BSP floorplan is mainly divided into the following two regions:

- **Static region**: Represents the region having BSP related hardware that remains static. The timing is closed for this region during base compilation. In general, the goal is to minimize the chip resources used by this region to close timing.
- **Kernel region**: Represents the partial reconfiguration (PR) region that is reserved for `freeze_wrapper_inst|kernel_system_inst` module, which contains the kernel. In general, the goal is to reserve chip resources to a maximum extent for this region.

---

(1) The Intel FPGA SDK for OpenCL is based on a published Khronos Specification, and has passed the Khronos Conformance Testing Process. Current conformance status can be found at www.khronos.org/conformance.

(2) OpenCL and the OpenCL logo are trademarks of Apple Inc. and used by permission of the Khronos Group™.

Guidelines for OpenCL BSP Floorplanning

- Begin with flat compilation to understand where all main components of the BSP gets placed naturally (especially the IP blocks with I/O connections such as PCIe® or DDR). While designing the BSP, you might have to consider establishing pipeline stages in between the IPs to close timing. You should first run a flat compile seed sweep to identify the recurrent failing paths, and then attempt to fix them.

  Tip: — A good timing closure rate over flat compile seed sweeps will have higher chances of closing base compile timing.

  — If you observe consistent failures in `mm_interconnect*` (component added by Qsys), then open the System with Qsys Interconnect viewer and observe the complexity of the failing interconnect. You can add pipelining flipflops in the viewer to improve timing. If you still cannot address the issue, you might have to break down the `mm_interconnect*` critical path by adding Avalon® pipeline bridges.

- During base compilation, start with LogicLock® on kernel region that contains `freeze_wrapper_inst|kernel_system_inst`. With no other restrictions, Intel Quartus® Prime can place the BSP hardware freely in the remaining static region of the chip. Use the flat compile and chip planner to identify the size and location of the BSP hardware, such as PCIe and DDR. Then, reserve the kernel region by using LogicLock while avoiding the main clustered areas of the BSP hardware.

  Tip: If the chip family used is same as the reference platform and if the BSP components are similar, it might be faster to start with the LogicLock regions for `freeze_wrapper_inst|kernel_system_inst` that is shipped with the OpenCL reference BSP and work through the failures.

- You might add the following additional components to your BSP:

  — **Memory banks**: If you add more memory banks, you should identify the I/O bank location since you may need to add pipeline bridges to meet timing.

  — **I/O channels**: You can add I/O channels such as video, Ethernet, or serial interface. If you add I/O channels, you should identify the I/O bank location since you might need to apply new LogicLock regions for pipelining if closing timing is difficult.

  Tip: If you need to add pipeline bridges (for example, due to large routing delays causing timing failures), then consider the routing distance from source to destination logic in the chip and release some space reserved for the kernel region.

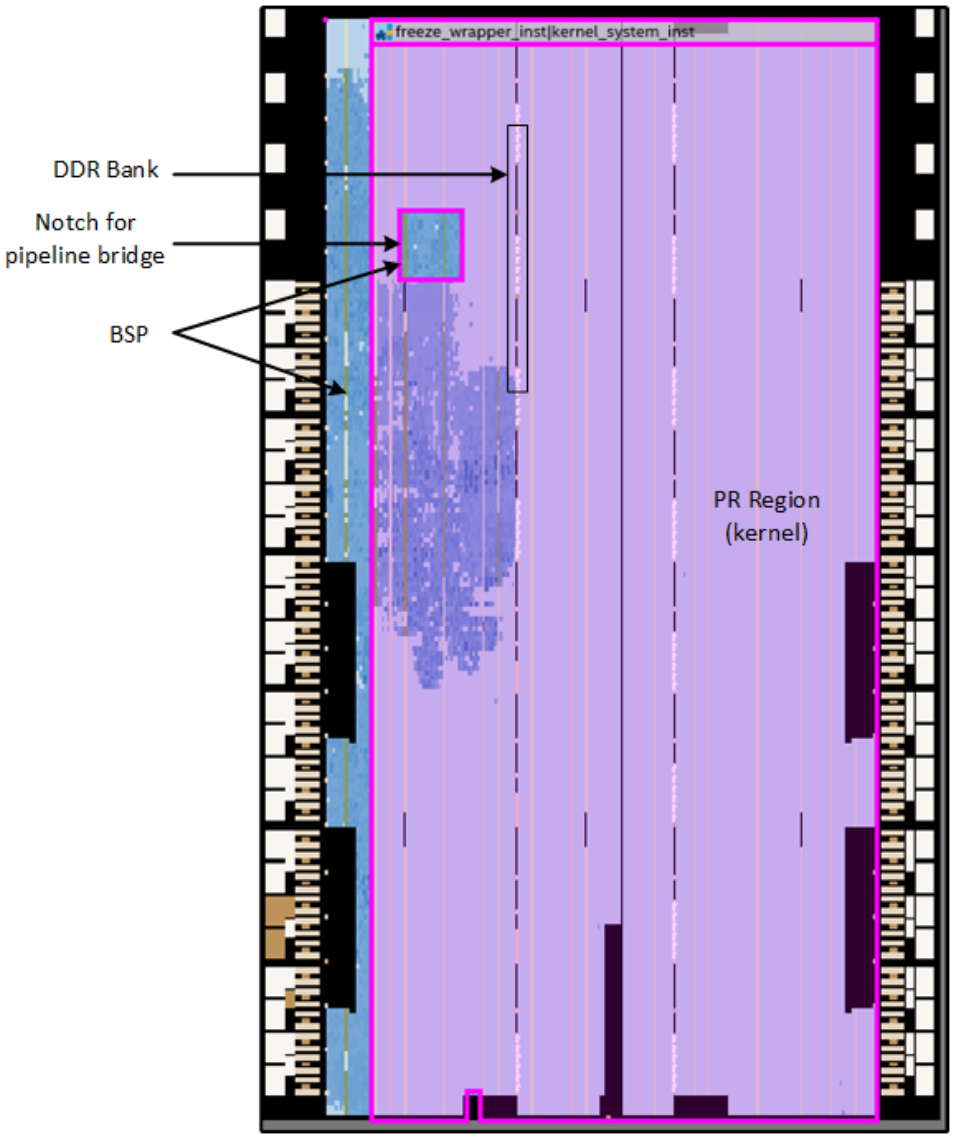
- Follow these general guidelines when reserving LogicLock regions for the kernel:

  — Attempt to place all DSP columns in the `kernel_system` unless required by the BSP.

  — Attempt to reserve more resources for the `kernel_system`.

  — Attempt to keep the number of notches in the kernel region to a minimum. The following figure illustrates a notch that was added to place a pipeline bridge between PCIe and DDR bank.

Figure 1. OpenCL BSP Floorplan for Intel Arria® 10 GX in the 17.0 Release



Guidelines for Maximum Operating Frequency

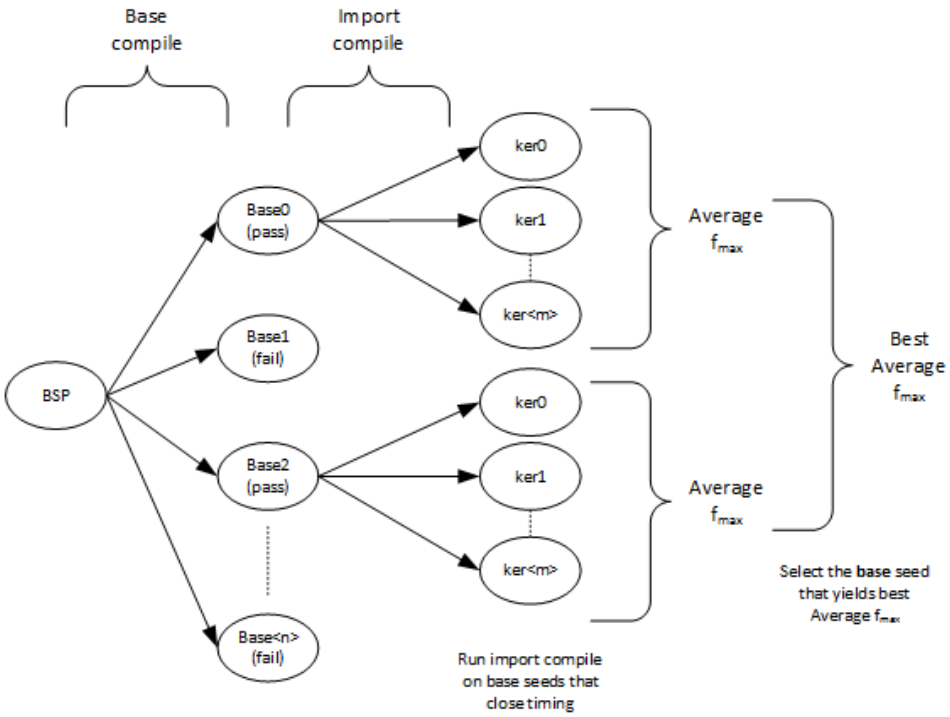
Maximum operating frequency (f_{max}) achieved by kernels largely depends on FPGA speed since most of the IPs should already be optimized . However, there might be some f_{max} loses depending on the BSP floorplan. For example, usually the number of cut-outs in the kernel region of BSP affects kernel f_{max}.

As illustrated in the following figure, to acquire the best base seed that yields the best average f_{max}:

1. Perform a seed sweep on the base compilation instead of selecting the first base seed that meets the timing.
2. Perform import compilation (by using few kernels from the example designs) on all the passing base seeds.
3. Compute the average f_{max} for all base seeds.
4. Select the base seed that yields the highest average f_{max}.

The base seed with the best average f_{max} is a good candidate for release with BSP. If you decide to follow an approach different than the recommended steps, you might observe 5-10% variation in the f_{max} of the kernel import compilation process.

Figure 2. Identifying the Best Base Seed



Tip:

- To understand how fast the kernel can run without floorplan restrictions:
  1. Perform a flat compilation of the kernel and observe the f_{max}.
  2. Perform an import compilation on the same kernel and observe the f_{max}.
  3. Compare f_{max} results.

  Due to the floorplan restrictions, import compile f_{max} is always lower than flat compile f_{max}. To avoid seed noise, compile the kernel with more base seeds and consider an average f_{max} while comparing f_{max} results.
- Never compare kernel f_{max} from a base compilation with a flat or an import compilation. Kernel clock targets are relaxed during base compilation and hence, you will never obtain good results.
- Observe the kernel clock critical path in base or import compilation. If the critical path is crossing from the kernel to the static region in the floorplan, change the floorplan or run few more base seeds to avoid this critical path.

Guidelines for Evaluating BSP Resource Utilization Efficiency

The higher the resource utilization percentage, the better the area utilization in the static area of your BSP. A high resource utilization percentage also implies that more resources are available for the kernel region.

Follow the steps below to calculate the resource utilization percentage of your BSP:

1. Obtain values for all resources in the FPGA from the `top.fit.rpt` or `base.fit.rpt` available under the Partition Statistics section of the Fitter report.
2. Deduct the value for "`freeze_wrapper_inst|kernel_system_inst`" (kernel region).

Tip: Focus more on the values of adaptive logic module (ALM) than on the values of other resources. Ensure that the resource utilization percentage for ALM is closer to the OpenCL reference BSP. A very high percentage for ALM might lead to congestion, which can increase the compilation time and introduce routing congestions in complex kernels. However, you can always increase or decrease the static region area, and observe the compilation time and f_{max}.

The following table reflects the OpenCL BSP resource utilization of Arria® 10 GX devices in the 17.0 release.

Table 1. OpenCL BSP Resource Utilization of IntelArria 10 GX devices in the 17.0 Release

	Total Available	Reserved for Kernel	Available for BSP	Used by BSP	%
ALM	427200	393800	33400	23817.5	71.31%
Registers	1708800	1575200	133600	38913	29.13%
M20K	2713	2534	179	134	74.86%
DSP	1518	1518	0	0	N/A

Observe that the floorplanning is executed in such a way that the static region will not have any DSP blocks.

Document Revision History

Table 2. Document Revision History of the Intel FPGA SDK for OpenCL Board Support Package Floorplan Optimization Guide

Date	Version	Changes
August 2017		Initial release.