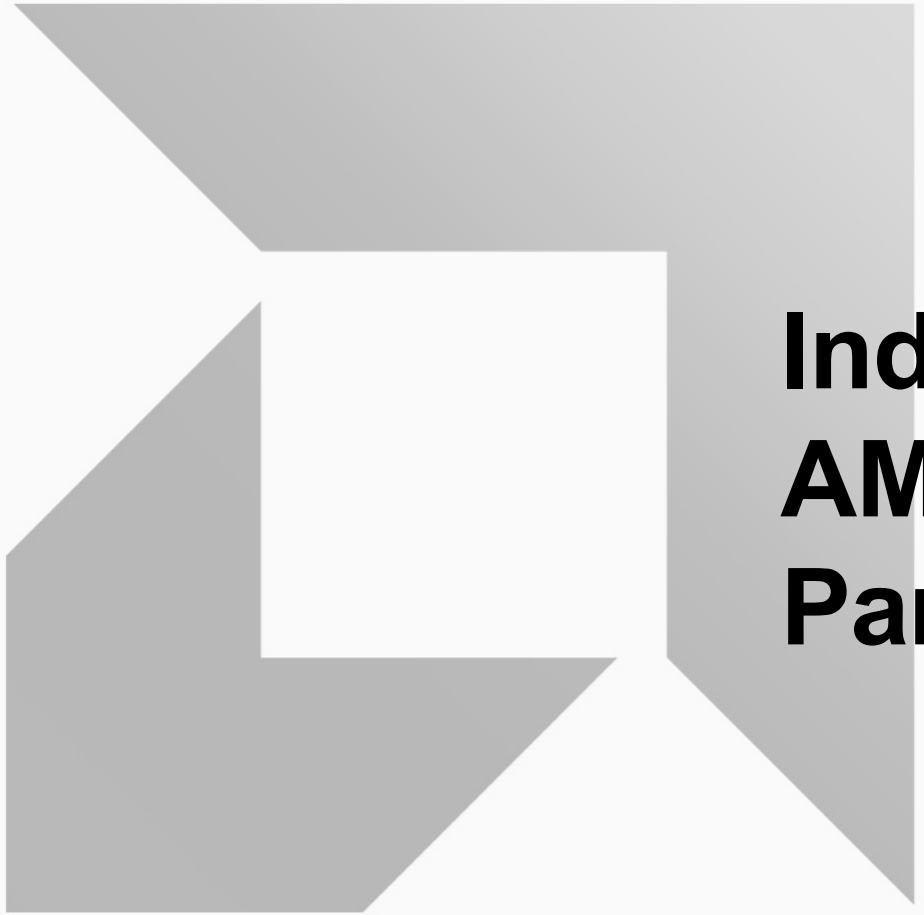


Independent Control of AMD Versal™ AI Engine Partitions

AMD
together we advance_

[Public]

Agenda

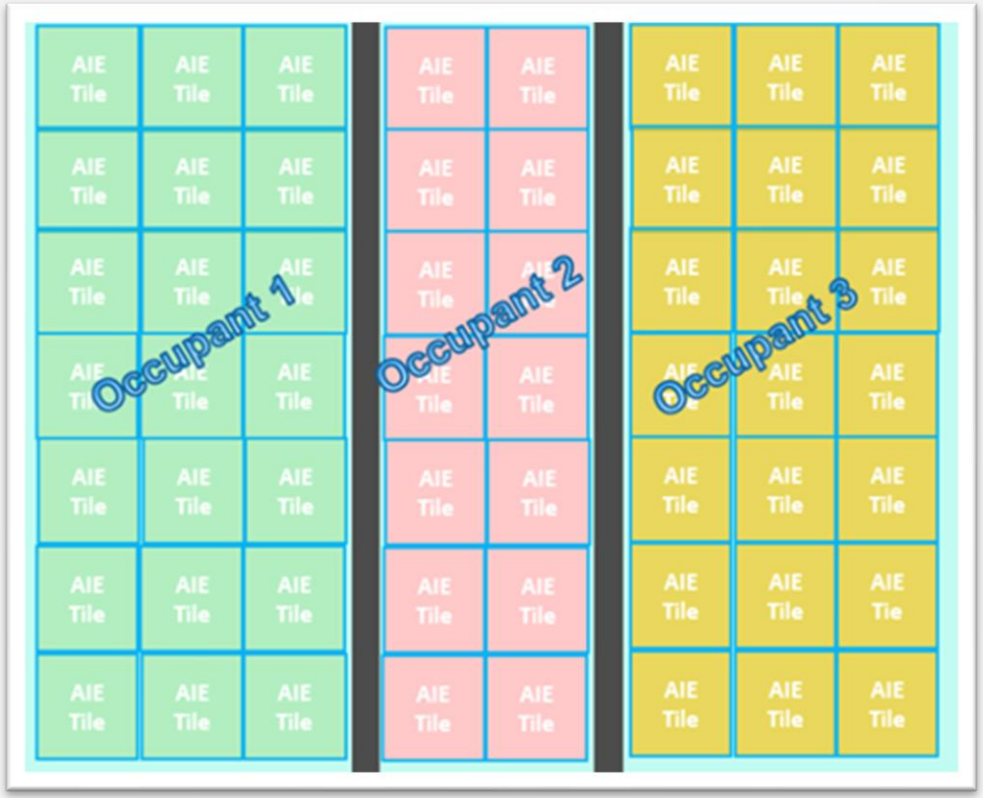
- **Introduction to AMD Versal™ AI Engine Independent Partitioning**
- AI Engine Kernel Workflow
- Packaging XCLBIN and Hardware Context
- Modular Control over Partition and Reloading
- Partition Modes and System Considerations
- Summary & Key Benefits

AMD
together we advance_

[Public]

Introduction to AMD Versal™ AI Engine Independent Partitioning

Versal™ AI Engine array can be partitioned to run independent partitions

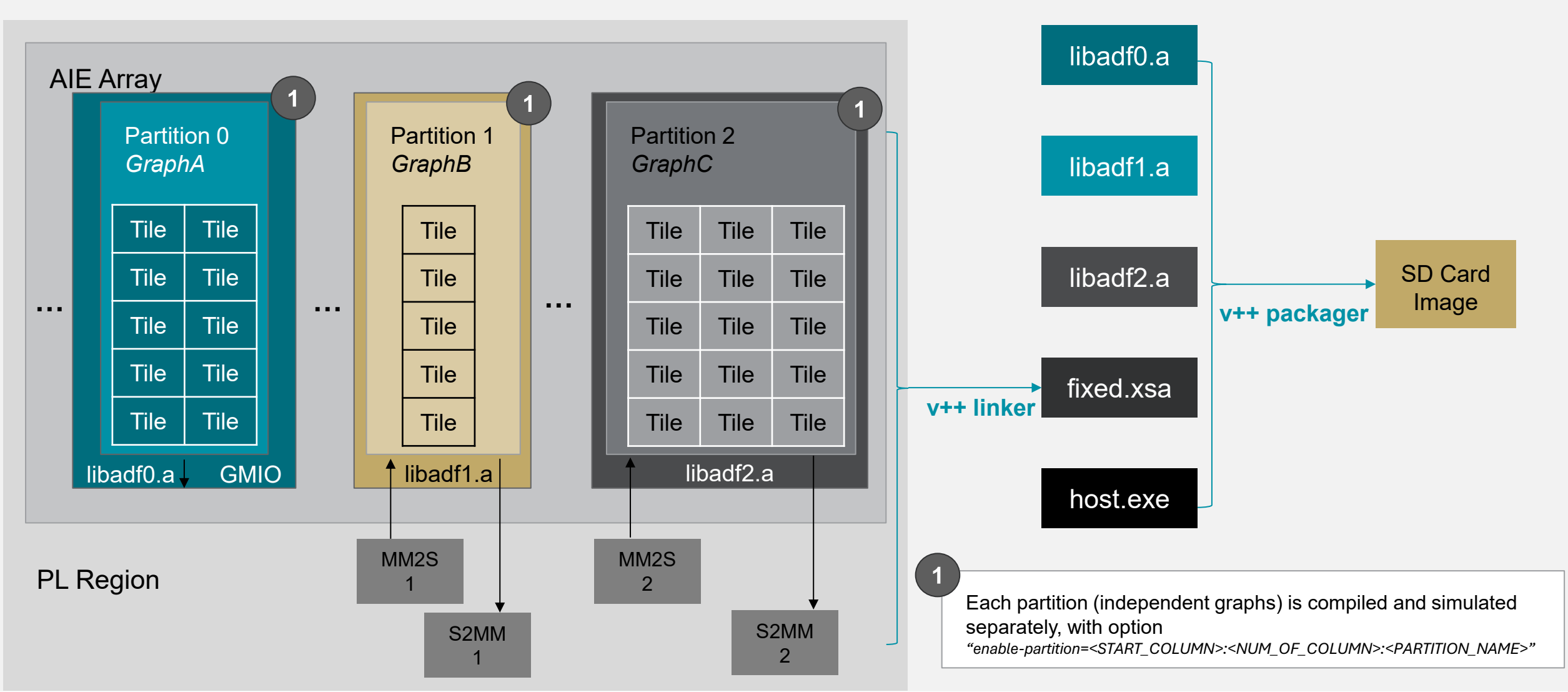


- Simplified compile/run workflow for each partition
- Modular capability; no need to recompile partitions
- Runtime control for flexible execution
- Partition based on application-specific needs
- Enables separate handling and control of each partition

AMD
together we advance_

[Public]

Introduction to Versal™ AI Engine Independent Partitioning



AMD
together we advance_

[Public]

Agenda

- Introduction to AMD Versal™ AI Engine Independent Partitioning
- **AI Engine Kernel Workflow**
- Packaging XCLBIN and Hardware Context
- Modular Control Over Partition and Reloading
- Supported Features and System Considerations
- Summary & Key Benefits

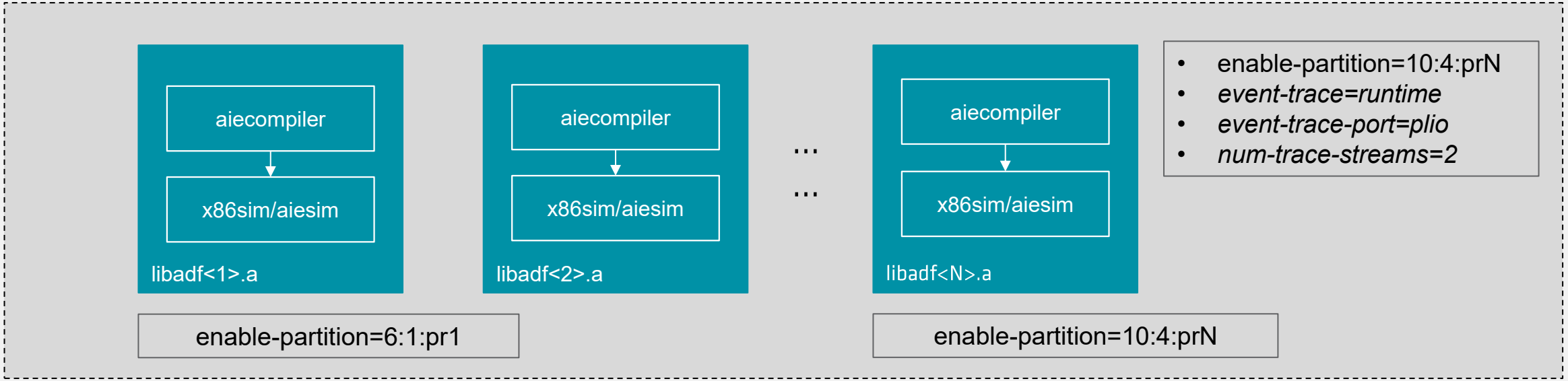
AMD
together we advance_

[Public]

AMD Versal™ AI Engine Kernel Workflow: Compilation

Define partitions: *enable-partition=<START_COLUMN>:<NUM_OF_COLUMN>:<PARTITION_NAME>*

Pass to the AMD Vitis™ command line: *v++ -c --mode aie --config <CONFIG_FILE.cfg>*



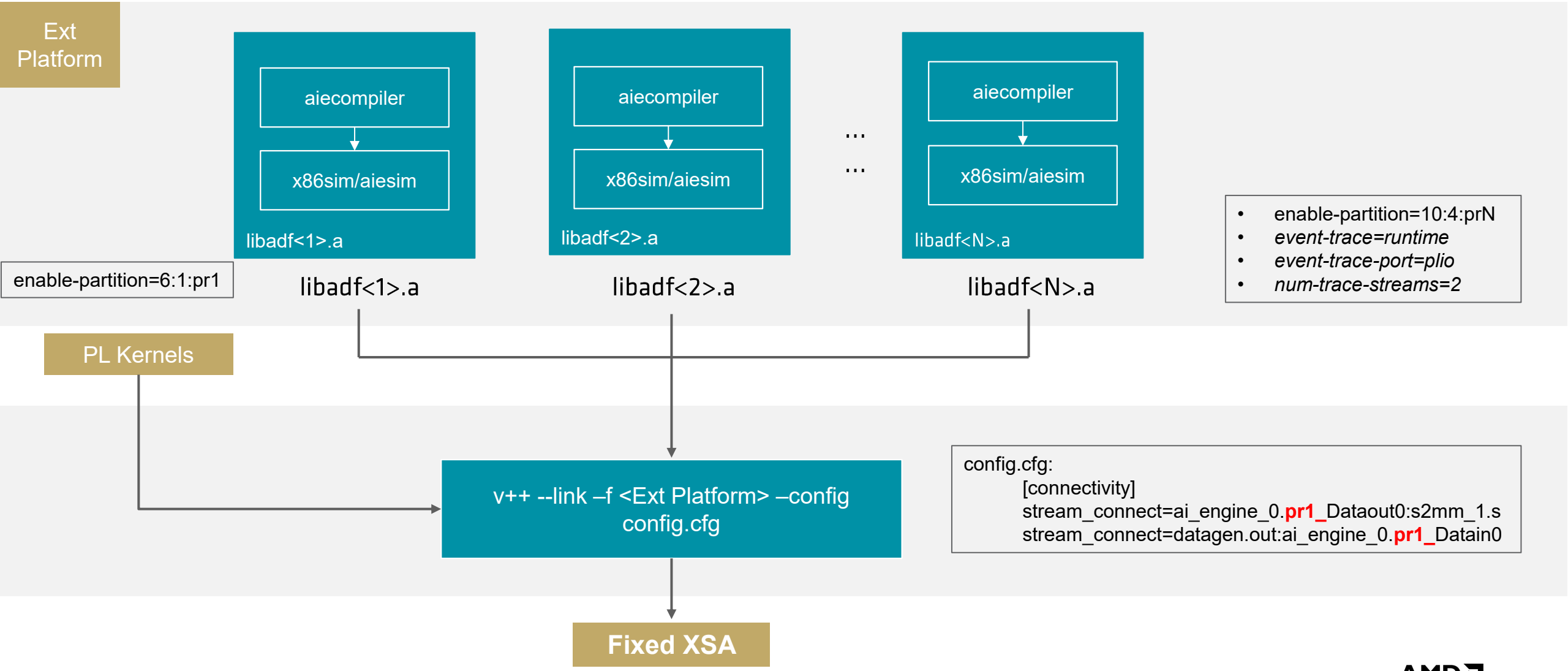
Example configuration (<CONFIG_FILE.cfg>):*[aie]*
enable-partition=6:2:pr0

This configuration restricts the compiled graph to columns 6 and 7, and names the partition

AMD
together we advance_

[Public]

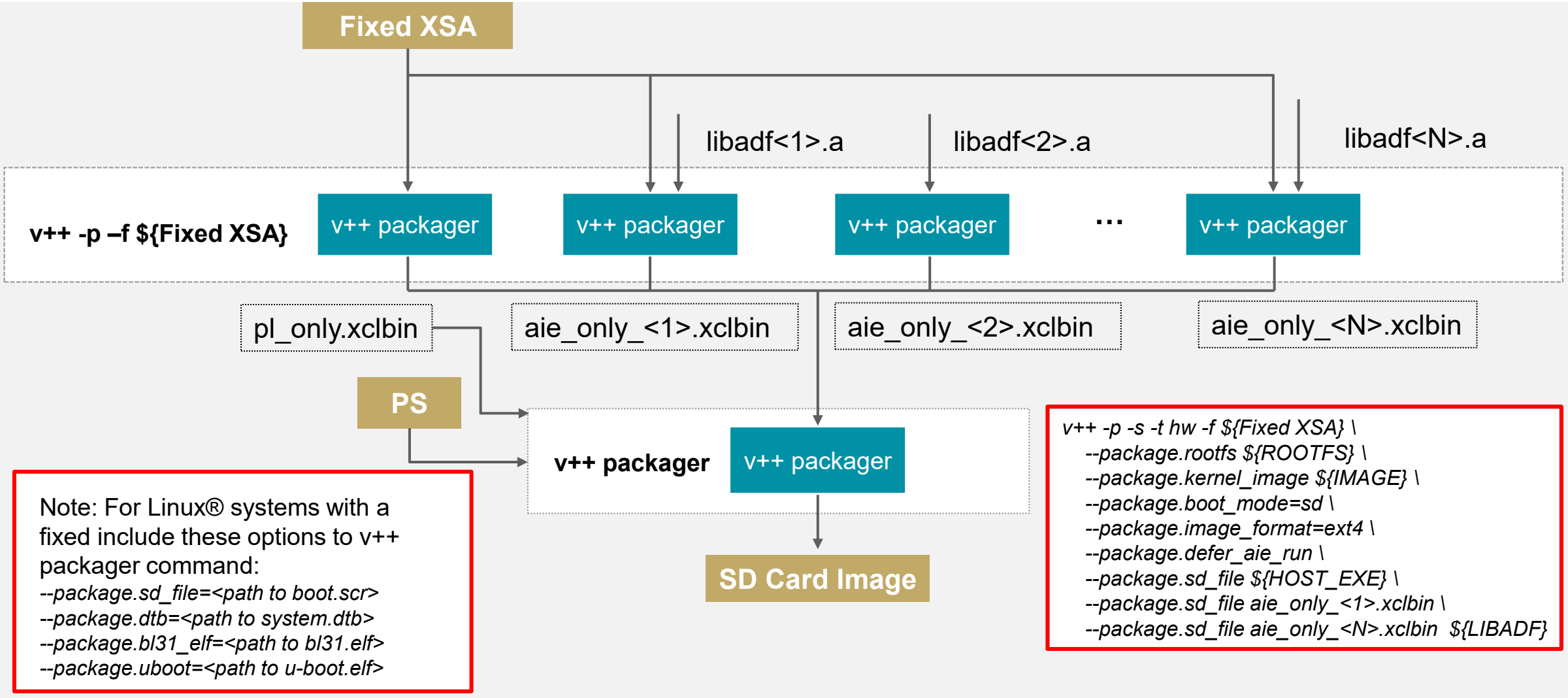
AMD Versal™ AI Engine Kernel Workflow: v++ Linker



AMD
together we advance_

[Public]

AMD Versal™ AI Engine Kernel Workflow: v++ Package



Note: For Linux® systems with a fixed include these options to v++ packager command:
--package.sd_file=<path to boot.scr>
--package.dtb=<path to system.dtb>
--package.bl31_elf=<path to bl31.elf>
--package.uboot=<path to u-boot.elf>

```
v++ -p -s -t hw -f ${Fixed XSA} \
   --package.rootfs ${ROOTFS} \
   --package.kernel_image ${IMAGE} \
   --package.boot_mode=sd \
   --package.image_format=ext4 \
   --package.defer_aie_run \
   --package.sd_file ${HOST_EXE} \
   --package.sd_file aie_only_<1>.xclbin \
   --package.sd_file aie_only_<N>.xclbin  ${LIBADF}
```


AMD
together we advance_

[Public]

Agenda

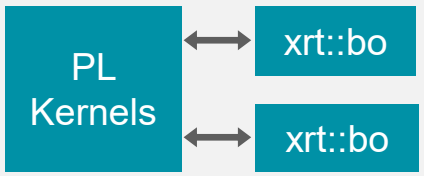
- Introduction to AMD Versal™ AI Engine Independent Partitioning
- AI Engine Kernel Workflow
- **Packaging XCLBIN and Hardware Context**
- Modular Control Over Partition and Reloading
- Partition Modes and System Considerations
- Summary & Key Benefits

AMD
together we advance_

[Public]

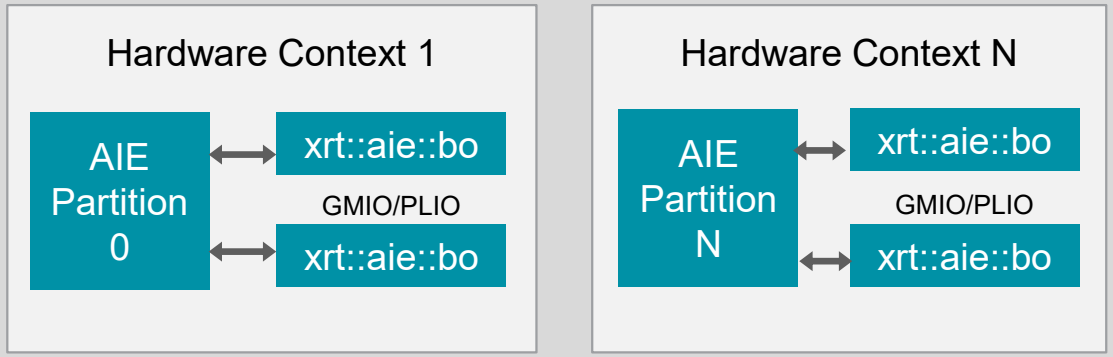
Packaging: XCLBIN & Hardware Context

XCLBINs are being packaged separately



Configuring a Flat XCLBIN for Programmable Logic (PL) Only

Uses a config (package_pl_only.cfg) to generate a *flat* XCLBIN containing only the programmable logic (PL) bitstream



Separate Packaging of AMD Versal™ AI Engine Overlays as Independent XCLBINs

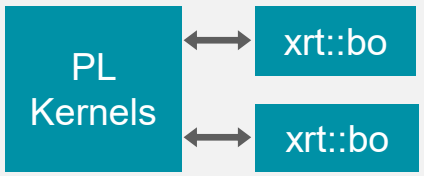
- Package each Versal AI Engine overlay (**libadf.a**) separately into independent XCLBINs
- Each XCLBIN corresponds to a specific partition, such as **pr1, pr2, ..., pr<N>**

AMD
together we advance_

[Public]

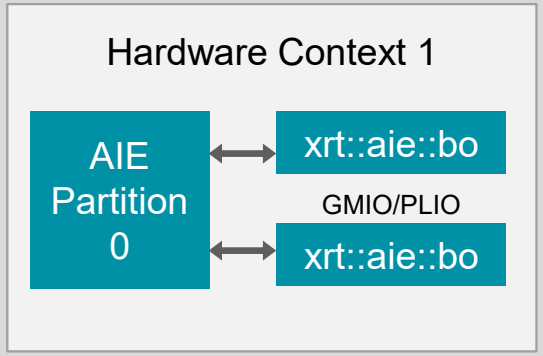
Packaging: XCLBIN & Hardware Context

XCLBINs are being packaged separately



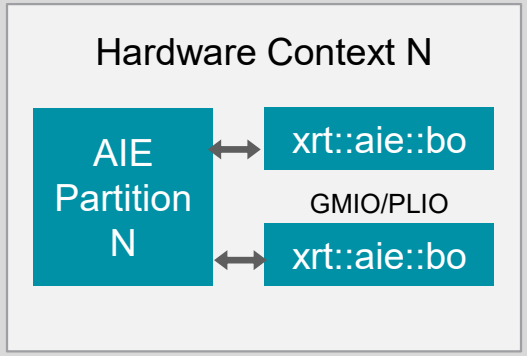
Package PL-only XCLBIN

- package_pl_only.cfg:
  [advanced]
  param=package.generateFlatPlVersalXclbin=1
- v++ packager command:
  *v++ -p -s -t hw -f ${FIXED XSA} *
  *--config package_pl_only.cfg *
  --output pl.xclbin



Package AMD Versal™ AI Engine-only XCLBIN 1

```
${VCC} -p -s -t hw -f ${FIXED XSA} \
        --package.defer_aie_run \
        --package.aie_overlay ./pr1/libadf.a \
        --output pr1.xclbin
```



Package Versal AI Engine-only XCLBIN <N>

```
${VCC} -p -s -t hw -f ${FIXED XSA} \
        --package.defer_aie_run \
        --package.aie_overlay ./pr<N>/libadf.a \
        --output pr<N>.xclbin
```


AMD
together we advance_

[Public]

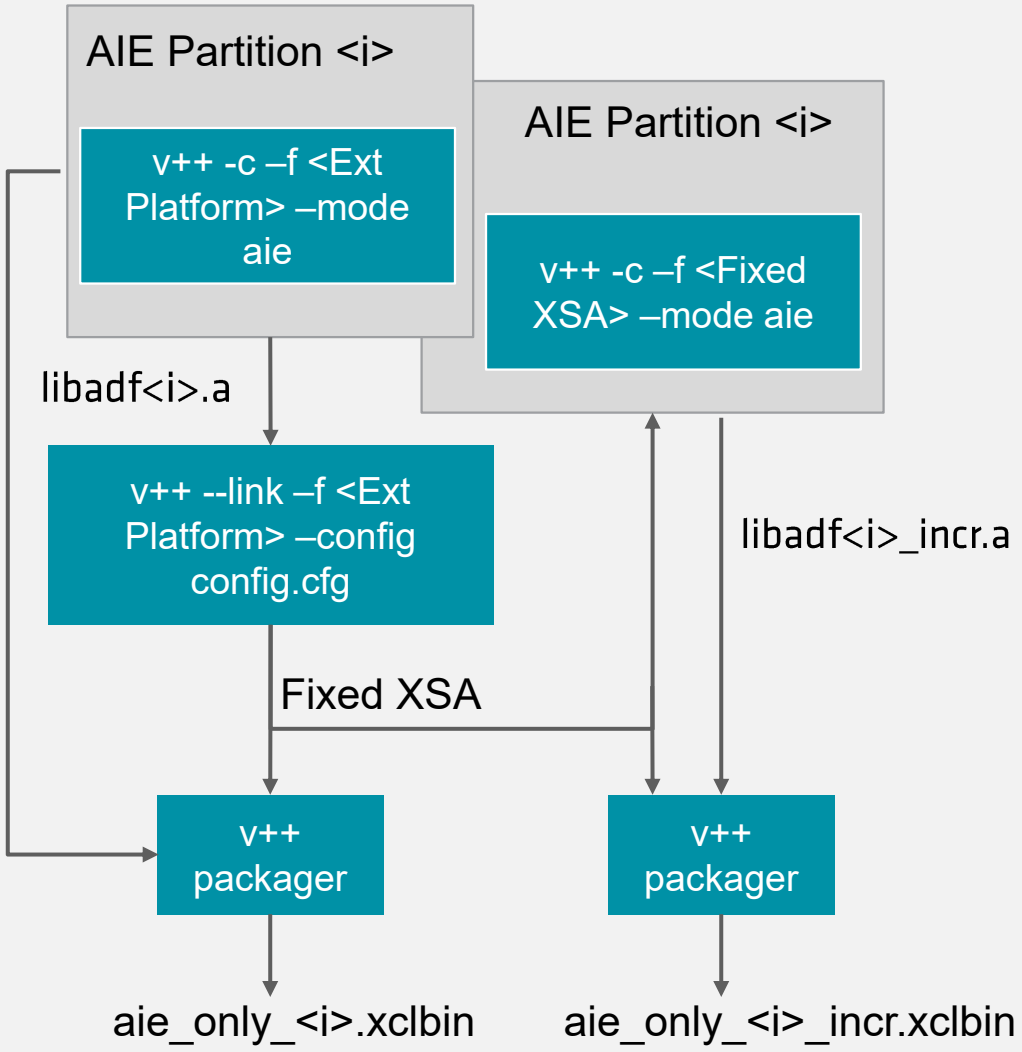
Agenda

- Introduction to AMD Versal™ AI Engine Independent Partitioning
- AI Engine Kernel Workflow
- Packaging XCLBIN and Hardware Context
- **Modular Control Over Partition and Reloading**
- Partition Modes and System Considerations
- Summary & Key Benefits

AMD
together we advance_

[Public]

Modular Control over Partition and Reloading



- Incremental AIE compilation:

  v++ -c --mode aie --platform=${FIXED XSA} --config aie.cfg ……
  --aie.output-archive=pr<i>/libadf<i>_incr.a ${GRAPH}

- Package AMD Versal™ AI Engine-only XCLBIN for the Partition

  ${VCC} -p -s -t hw -f ${FIXED XSA} \
  --package.defer_aie_run \
  --package.aie_overlay ./pr<i>/libadf<i>_incr.a \
  --output pr<i>_incr.xclbin

Faster build cycles

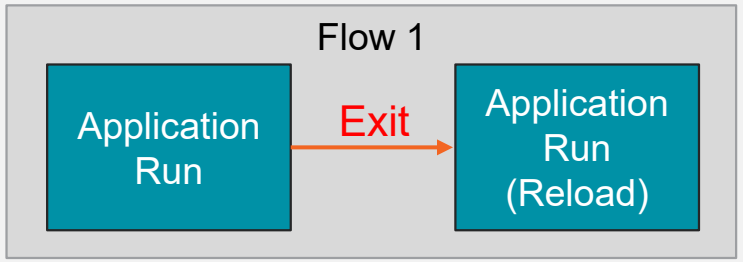
Modular updates

Runtime control

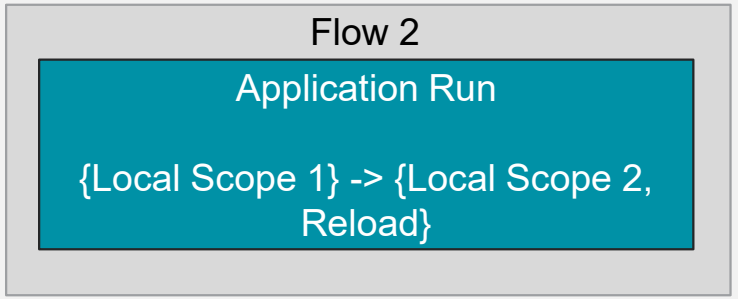
AMD
together we advance_

[Public]

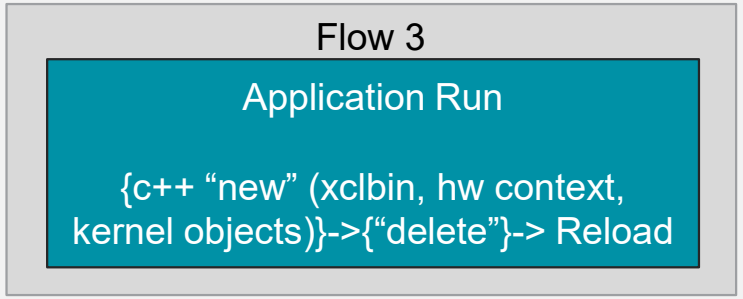
Modular Control over Partition and Reloading



Flow 1:
- After application run, exit the application and run again to reload the partition



Flow 2 with Partition Reload:
- Execute AI Engine graph for a specific partition
- Wait for PL kernels and graphs to finish
- Reload only the targeted partition while keeping the rest of the system intact

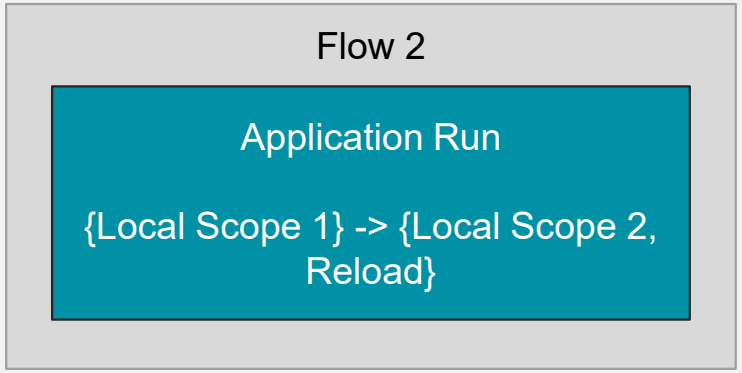


Flow 3:
- Use c++ “new” to create xclbin, hardware context, graph, and kernel objects, and delete these objects before reloading the xclbin

AMD
together we advance_

[Public]

Modular Control over Partition and Reloading



First execution (Local Scope 1)

```
auto xclbin_aie_pr2 = xrt::xclbin(xclbinFilename_aie_pr2);
auto uuid_aie_pr2 = device.register_xclbin(xclbin_aie_pr2);
xrt::hw_context hwctx_aie_pr2{device, uuid_aie_pr2};
auto ghdl=xrt::graph(hwctx_aie_pr2,"gr"); //"gr" is graph object name
ghdl.run(iterations);
```

Local scope 2 for reloading the partition

```
  auto xclbin_aie_pr2_2 = xrt::xclbin(xclbinFilename_aie_pr2_2);
  auto uuid_aie_pr2_2 = device.register_xclbin(xclbin_aie_pr2_2);
  xrt::hw_context hwctx_aie_pr2{device, uuid_aie_pr2_2};
// Launch graph in the AI Engine partition
  auto ghdl=xrt::graph(hwctx_aie_pr2,"gr"); //"gr" is graph object name
  ghdl.run(iterations);
```

#Before reloading, ensure PL kernels are not sending data to the partition. Ensure PS is not updating other AMD Versal™ AI Engine partitions when reloading.

AMD
together we advance_

[Public]

Agenda

- Introduction to AMD Versal™ AI Engine Independent Partitioning
- AI Engine Kernel Workflow
- Packaging XCLBIN and Hardware Context
- Modular Control Over Partition and Reloading
- **Partition Modes and System Considerations**
- Summary & Key Benefits

AMD
together we advance_

[Public]

Partition Modes and System Considerations

Single vs Multi Partition of AMD Versal™ AI Engine Array

Single Partition Configuration

Entire array as a single partition

When to Use

- Independent reloading of graphs not required
- AI Engine array resources are at a premium and must be optimized
- No need to track the individual AI Engine xclbin partition images

Multi Partition Configuration

Array partition into smaller, columnar sections

When to Use

- Multiple graphs require independent reloading
- Resource usage is not a primary concern
- Requires AI Engine graph debugging during early design phases

AMD
together we advance_

[Public]

Partition Modes and System Considerations

Key Considerations For Managing Independent Multi-Partition Configurations

- AMD Versal™ AI Engine partitioning is column-based, with resources exclusive to each partition, potentially causing performance inefficiencies.
- When reloading a partition:
  - Ensure PL does not send data via AXIS or interact with the Versal AI Engine through AXIMM
  - Manually reset the Vitis Region and all PL IPs connected to Versal AI Engine
  - Flush any in-transit data samples in PL
- Ensure that PS does not interact with other Versal AI Engine partitions when reloading partition.
- Reloading affects only the Versal AI Engine partition; system-level error recovery is the user's responsibility.
- AMD Vitis™ Analyzer supports summary or trace for one partition at a time.
- HDL+AIE co-simulation is not supported when partitions are enabled.
- Bare-metal support for these features is limited or in beta.

AMD
together we advance_

[Public]

Agenda

- Introduction to AMD Versal™ AI Engine Independent Partitioning
- AI Engine Kernel Workflow
- Packaging XCLBIN and Hardware Context
- Modular Control Over Partition and Reloading
- Partition Modes and System Considerations
- **Summary & Key Benefits**

AMD
together we advance_

[Public]

Summary & Key Benefits

01	Independent control of each AMD Versal™ AI Engine partition allows for better resource management
02	Partition reloading targets just the partition, ensuring minimal disruption to the overall system
03	Column-based partitioning ensures that partitions do not overlap in resources
04	Event-tracing for single or all partitions at runtime improves debugging and performance monitoring
05	Users have control over partition reloads and are responsible for system-level error recovery

AMD
together we advance_

[Public]

General Disclaimer and Attribution Statement

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18u.

© 2025 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, Versal, Vitis, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries. Other product names used in this publication are for identification purposes only and may be trademarks of their respective owners. Certain AMD technologies may require third-party enablement or activation. Supported features may vary by operating system. Please confirm with the system manufacturer for specific features. No technology or product can be completely secure.


AMD