logic fruit Technologies

WHITEPAPER

ACCELERATED

Application development on AMAZON EC2 F1

By :
Dheeraj Punia And Daivik Bhatia

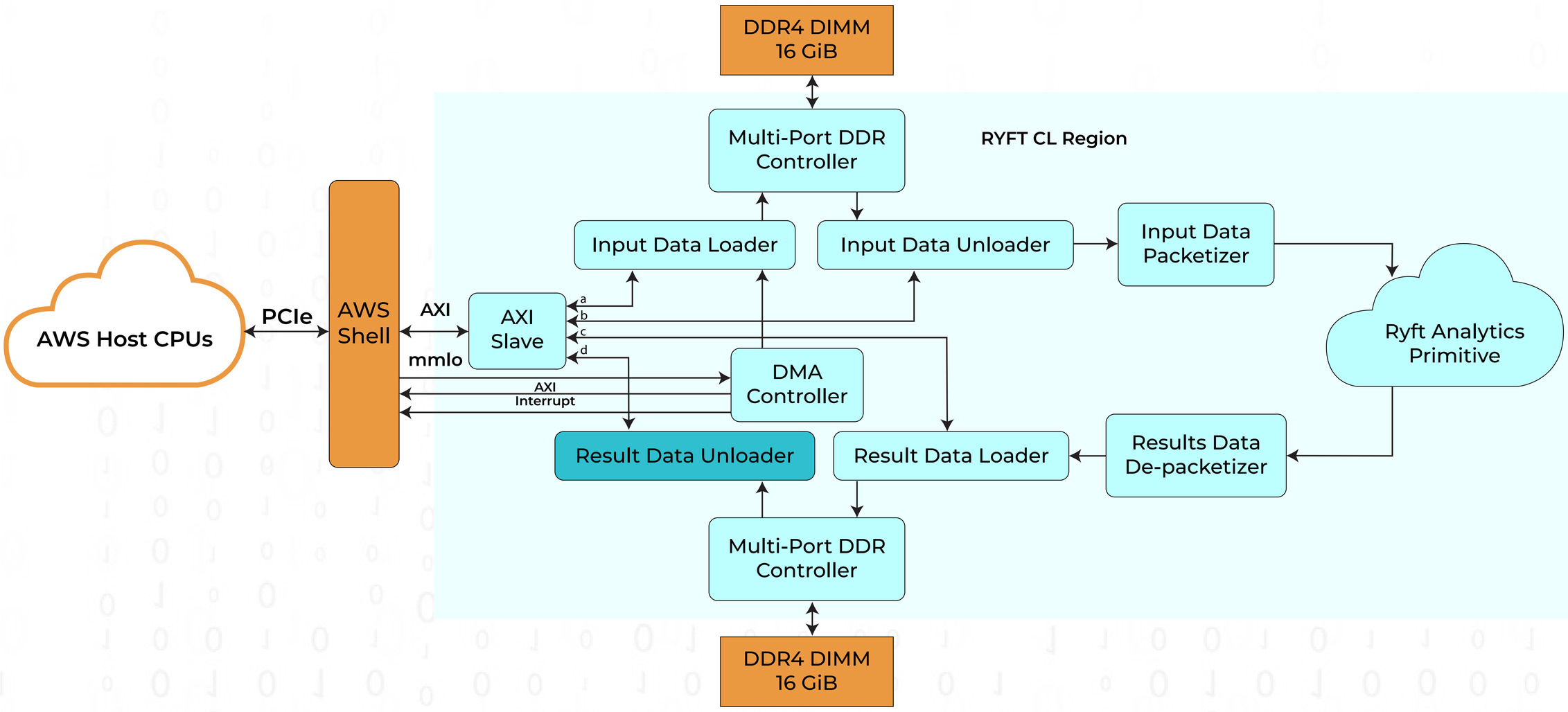
OVERVIEW OF AWS EC2 FPGA DEVELOPMENT KIT

The AWS FPGA Development Kit is a set of development and runtime tools for designing, debugging, compiling, and running hardware-accelerated programs on Amazon F1 instances. The kit, which is available on *GitHub*, includes all documentation on F1, internal FPGA interfaces, and compiler scripts for producing **Amazon FPGA Images (AFIs)**.

Ethernet frames can be streamed from a network interface to the FPGA on F1 instances for processing and returned using the *Virtual Ethernet framework*, which supports shell versions F1.X.1.4 and F1.S.10. With no development tool fees, it's shared across this GitHub repository and AWS's FPGA Developer AMI - Centos/AL2.

Amazon EC2 F1 Instances

F1 instances are effective for a wide spectrum of developers, **from low-level hardware designers to software engineers that prefer C/C++ and openCL** environments. Once your FPGA design is complete, you can register it as an Amazon FPGA Image (AFI) and deploy it to your F1 instance in just a few clicks. Target applications that potentially benefit from F1 instance acceleration include genomic analysis, search/analytics, image, and video processing, network security, electronic design automation (EDA), image and file compression, and big data analytics.



In *Amazon EC2 F1 instances*, FPGAs are used to provide bespoke hardware accelerations. F1 instances are easy to program and come with everything you need to create, simulate, debug, and compile hardware acceleration code, including an FPGA Developer AMI and cloud-based hardware development.

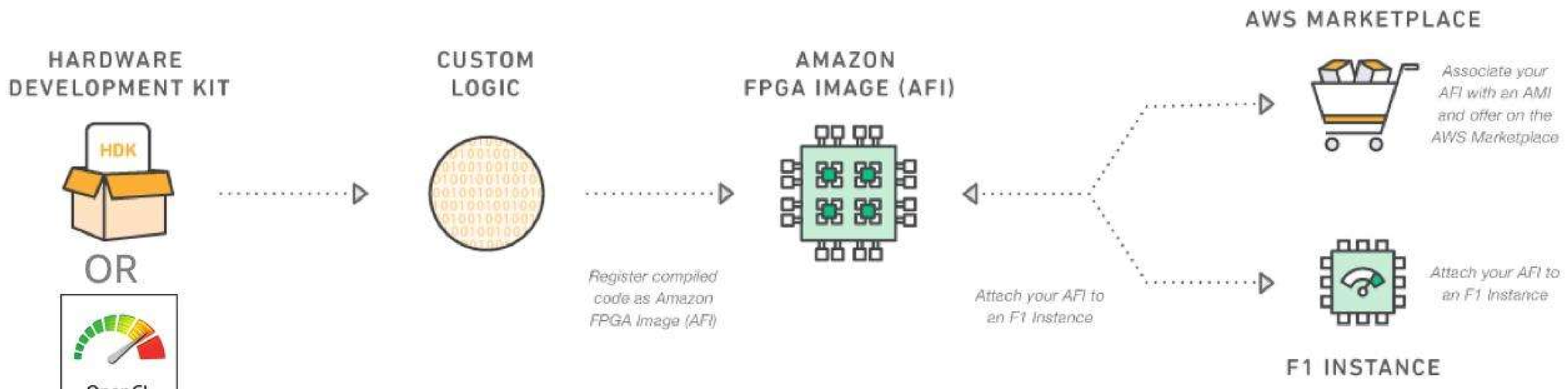
In a variety of applications, using F1 instances to install hardware accelerations can help overcome difficult research, engineering, and commercial difficulties that require high bandwidth, better networking, and extremely high computation capabilities. Some of the platform's most important features are given here; to learn more about F1 Instances, *click here.*

F1 Platform Features

- 1-8 Xilinx UltraScale+ VU9P based FPGA slots.
- User-defined clock frequency driving all CL to Shell interfaces.
- Multiple free-running auxiliary clocks.
- PCI-E endpoint presentation to Custom Logic(CL).
- Virtual JTAG, Virtual LED, Virtual DIP Switches.
- PCI-E interface between Shell(SH) and Custom Logic(CL).
- DDR interface between SH and CL.

Development Flow

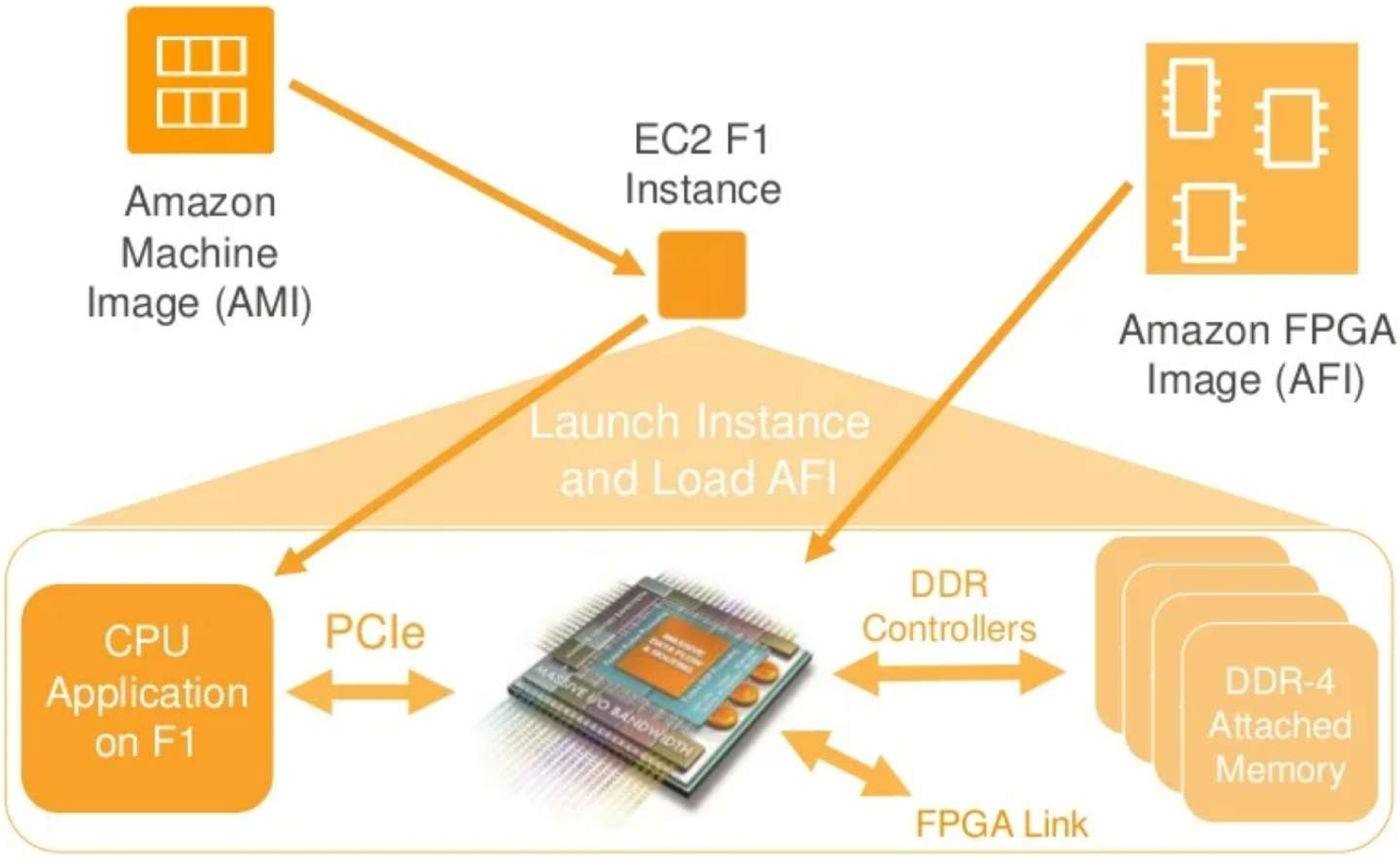
After building an FPGA design, developers can create an Amazon FPGA Image (AFI) and deploy it to an F1 instance (also known as CL - Custom logic). AFI is scalable and secure, and it can be reused, shared, and deployed. ***Click here*** for more information on the various development environments.



FPGA Developer AMI

The ***FPGA Developer AMI*** is available for free on the ***AWS marketplace***, and it includes tools for creating ***FPGA designs*** that run on AWS F1. Due to the large size of the FPGA utilised inside AWS F1 Instances, Xilinx tools function best with 32GiB Memory. The z1d.xlarge/c5.4xlarge and z1d.2xlarge/c5.8xlarge instance types would provide the fastest execution speed with 30GiB+ and 60GiB+ of RAM, respectively.

Developers looking to save money can begin coding and running simulations on low-cost instances such as t2.2xlarge, then shift their accelerated code synthesis to the larger instances. Depending on your needs, choose the appropriate specs for your instances. Planning ahead of time could save you a lot of time and money in the long run. The image below depicts how the AMI interface is used with the F1 instance for FPGA acceleration on AWS.



Hardware Development Kit (HDK)

Go to the ***HDK directory***, which contains documentation, examples, simulation, build, and AFI generating scripts, to get started creating Amazon FPGA Images. Installing the HDK on a local server or an Amazon EC2 instance is possible. **The development kit is not required if you plan to utilise a pre-built AFI provided by another developer.**

AWS Shells

Each FPGA instance on Amazon EC2 is separated into two partitions:

Shell Name	Shell Version	Dev Kit Branch	Description
F1 XDMA Shell	F1.X.1.4	*master*	Provides all the interfaces *listed here*, includes DMA.
F1 Small Shell	F1.S.1.0	*small_shell*	All of the interfaces *listed here* are available. This shell is devoid of the DMA engine, resulting in a considerable reduction in Shell resource consumption.

When the Shell and CL are combined at the end of the development process, an Amazon FPGA Image (AFI) is created that can be loaded onto Amazon EC2 FPGA Instances.

Software-defined Development Environment

Customers can use the software-defined development environment to compile their C/C++/OpenCL code into FPGA kernels and then use OpenCL APIs to transport data to the FPGA. Software developers will find a familiar programming environment that supercharges cloud applications even if they have no prior experience with FPGAs. This platform allows C/C++ and RTL accelerator designs to be integrated into a C/C++ program development environment.

This method enables faster prototyping in C/C++ while also permitting manual ***RTL optimization*** of key blocks. This approach is similar to using software compiler optimization techniques to improve time-critical routines.

Runtime Tools (SDK)

The ***SDK directory*** contains the runtime environment required to run on EC2 FPGA instances. It includes all of the essential drivers and tools for managing the AFIs loaded on the FPGA instance. The SDK is only required after an AFI has been installed on an EC2 FPGA instance; it is not necessary during the construction of an AFI. ***Click here*** to learn more about the SDK resources available.

ACCELERATE YOUR C/C++ APP ON AN AWS F1 FPGA INSTANCE WITH XILINX VITIS

The software-defined development flow is used to accomplish this. Xilinx Vitis and SDAccel is a complete development environment for applications that use Xilinx FPGAs. It employs the OpenCL heterogeneous computing architecture to offload **compute-intensive workloads to the FPGA**. The accelerated program is written in C/C++, OpenCL, or RTL using OpenCL APIs. Before continuing, double-check that you meet all of the ***Prerequisites.***

Build the host application and the Xilinx FPGA binary

Emulate your Design

Emulation's primary goal is to ensure that an application is functionally correct before determining how to partition it between the host CPU and the FPGA.

Software (SW) Emulation

For CPU-based (SW) emulation, both the host code and the FPGA binary code are compiled to execute on an x86 processor. ***SW Emulation*** allows engineers to iterate and refine algorithms quickly thanks to its rapid compilation. Iteration takes about the same amount of time as compiling and running software on a computer.

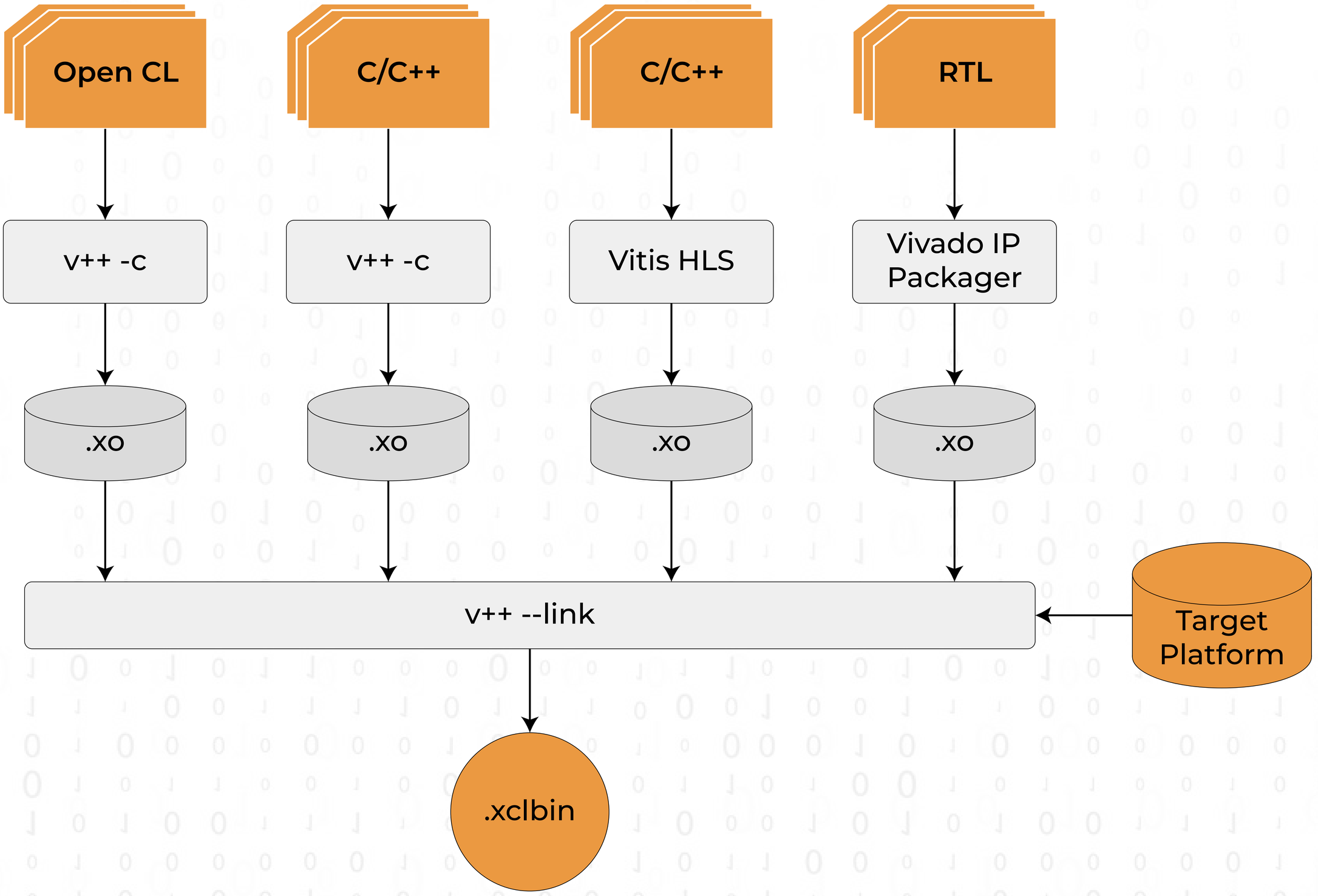
Hardware (HW) Emulation

The SDAccel ***hardware emulation*** cycle enables the developer to validate the logic generated for the FPGA binary. In this emulation cycle, the SDAccel hardware simulator is used to validate the functionality of the code that will be executed on the FPGA Custom Logic.

Generating xclbin

The ***host program***, which is written in C/C++ and employs either the ***XRT native API*** or ***OpenCLTM API*** calls, is built using the GNU C++ compiler (g++), which is part of the GNU compiler collection (GCC). Each source file is compiled into an object file (.o) and linked with the Xilinx® runtime (XRT) shared library to create the executable that runs on the host CPU.

As shown in the below diagram, the kernel code is written in C, C++, OpenCLTM C, or RTL, and it is built by compiling it into a Xilinx® object (XO) file and linking the XO files into a ***Xilinx binary*** (.xclbin) file.



Creating Amazon FPGA Image (AFI)

The FPGA (field-programmable gate array) AMI is a supported and updated CentOS Linux image from Amazon Web Services. All of the FPGA creation and run-time tools needed to construct and operate custom FPGAs for ***hardware acceleration*** are pre-installed on the AMI.

Setup CLI and Create S3 Bucket

The developer must create an S3 bucket for the AFI generation. The bucket will include a tar file as well as logs generated by the AFI creation service. The JSON output format is required by the AWS SDAccel scripts, and any other output format will cause the scripts to fail (ex: text, table). JSON is the default output format for the ***AWS CLI.***

$ aws configure # to set your credentials (found in your console.aws.amazon.com page), region (us-east-1) and output (json)

The AWS scripts will upload your DCP to AWS for AFI generation, which will be packaged into a tar file, using this S3 bucket. Begin by making a bucket:

$ aws s3 mb s3://<bucket-name> --region us-east-1 # Create an S3 bucket (choose a unique bucket name)

$ touch FILES_GO_HERE.txt # Create a temp file

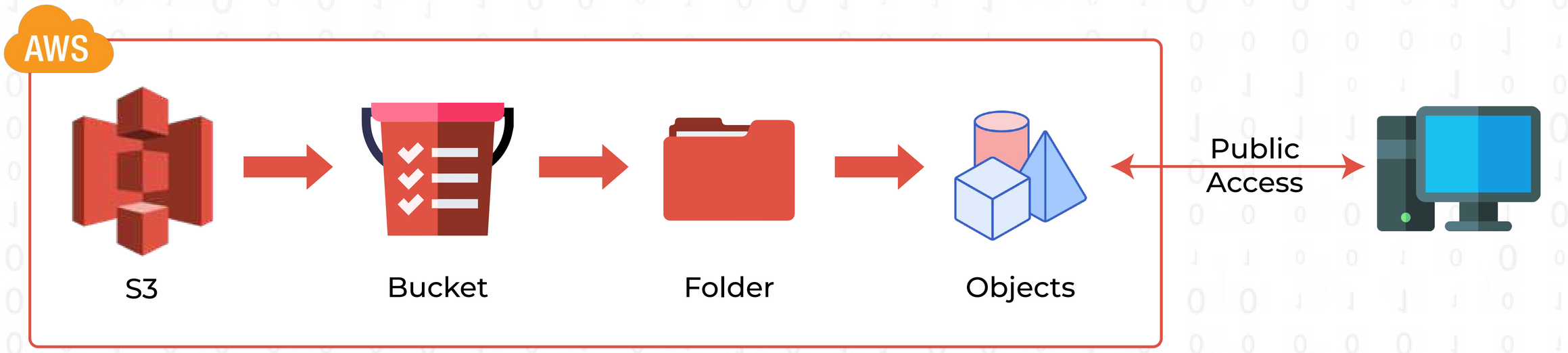
$ aws s3 cp FILES_GO_HERE.txt s3://<bucket-name>/<dcp-folder-name>/ # Choose a dcp folder name

During the AFI setup process, logs will be generated and stored in your S3 bucket. These logs could be used to debug if the AFI generation fails. Next, create a folder for your log files:

$ touch LOGS_FILES_GO_HERE.txt # Create a temp file

$ aws s3 cp LOGS_FILES_GO_HERE.txt s3://<bucket-name>/<logs-folder-name>/ # Choose a logs folder name

Once your AFI has been properly built, you can delete the tar file and logs as needed. If you delete these files, your AFI will not be affected.



You've now completed the build of your Xilinx FPGA Binary (xclbin) as well as the necessary CLI and S3 bucket. You are now ready to begin building the AFI. To learn how to *track the status of your registered AFI*, go here.

The *create sdaccel afi.sh* script is included to help you generate an AFI from a Xilinx FPGA binary. Here's how it works:

- ☞ Takes in your Xilinx FPGA Binary *.xclbin file
- ☞ Calls aws ec2 create_fpga_image to generate an AFI under the hood
- ☞ Generates a <timestamp>_afi_id.txt which contains the identifiers for your AFI
- ☞ Creates an AWS FPGA Binary file with an *.awsxclbin extension that is composed of Metadata and AGFI-ID.
  - ✓ This *.awsxclbin is the AWS FPGA Binary file that will need to be loaded by your host application to the FPGA

$ $ SDACCEL_DIR/tools/create_sdaccel_afi.sh
-xclbin=<input_xilinx_fpga_binary_xclbin_filename>
**-o=<output_aws_fpga_binary_awsxclbin_filename_root> **
-s3_bucket=<bucket-name> -s3_dcp_key=<dcp-folder-name>
-s3_logs_key=<logs-folder-name>

Run the FPGA accelerated application on Amazon F1 instances

Before starting an FPGA instance with the *FPGA Developer AMI* from AWS Marketplace, check the *AMI compatibility table* and the *runtime compatibility table*. To run your SDAccel programs, you can create *your own Runtime AMI* for Amazon FPGA instances. If the developer flow (compilation) was done on a different instance, you'll need to accomplish the following:

- ✓ Copy the compiled host executable (exe) to the new instance.
- ✓ Copy the *.awsxclbin AWS FPGA binary file to the new instance.
- ✓ Depending on the host code, the *.awsxclbin may need to named <hostcodename>.hw.<platformname>.awsxclbin
- ✓ Copy any data files required for execution to the new instance.
- ✓ Clone the *GitHub repository* to the new F1 instance and install runtime drivers.

$ git clone https://github.com/aws/aws-fpga.git $AWS_FPGA_REPO_DIR
$ cd $AWS_FPGA_REPO_DIR
$ source sdaccel_setup.sh

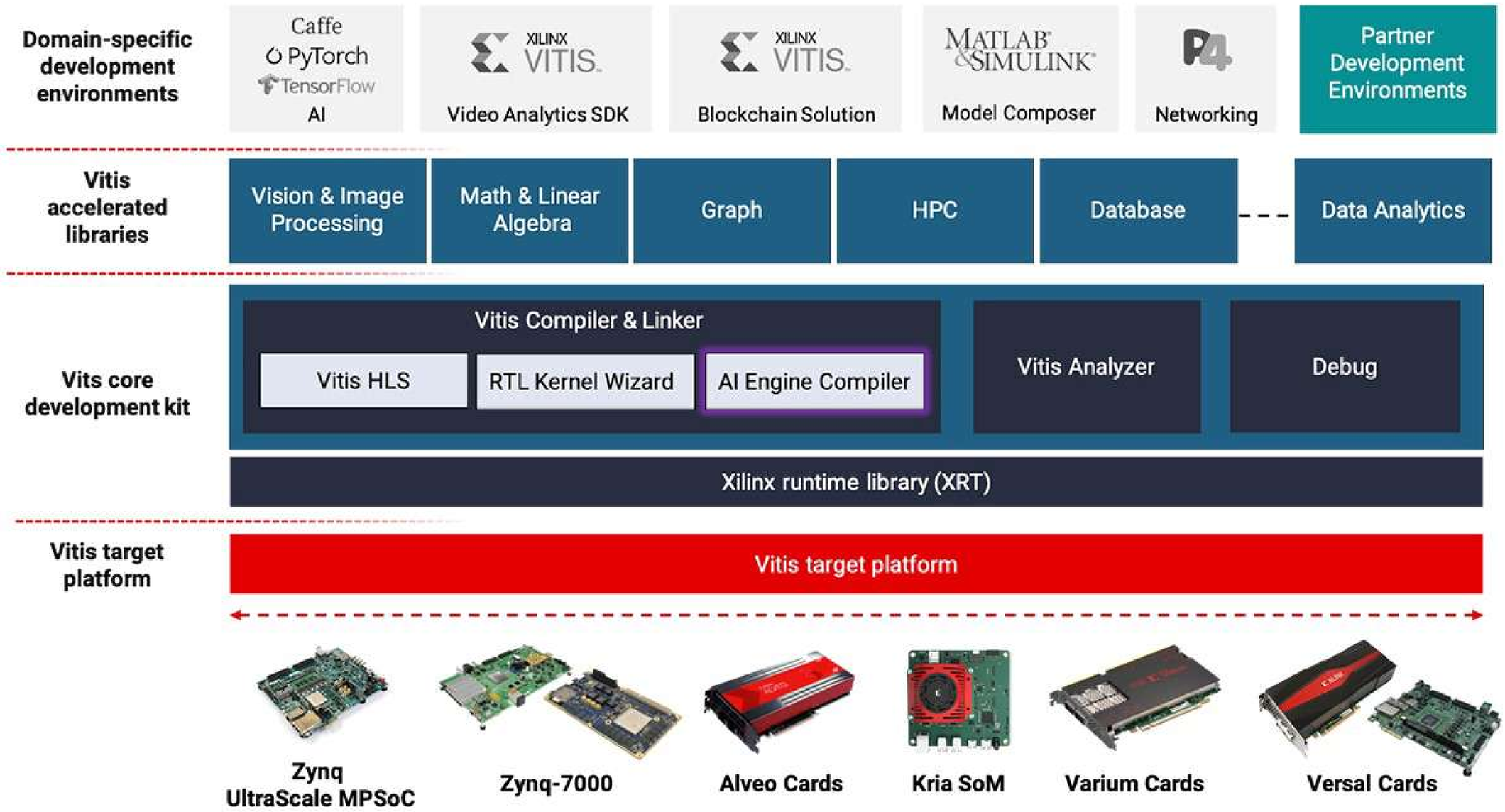
Ensure the host application can find and load the *.awsxclbin AWS FPGA binary file.

Source the Runtime Environment & Execute your Host Application:

$ sudo -E /bin/bash # source $AWS_FPGA_REPO_DIR/sdaccel_runtime_setup.sh # Other runtime env settings needed by the host app should be setup after this step # ./helloworld

Migration of Alveo U200 Design to AWS F1 Instance

The Vitis development flow provides the developer with platform-independent APIs and interfaces. This makes moving applications between similar FPGA acceleration cards a lot easier. The source code for the software application and the FPGA kernels remains intact in this development sequence. To port the application from Alveo U200 to AWS F1, just command line changes are required. ***This example*** shows how to move a Vitis application from an ***Alveo U200*** card to an AWS EC2 F1 instance.



Build for AWS F1

The only change necessary to migrate the accelerated application from Alveo U200 to AWS F1 is in the options. cfg file (configuration file). **The source code is unchanged, and the g++ and v++ commands are the same.**

1. Go to the f1 directory.
2. The options.cfg file for AWS F1 contains the following options:

   platform=xilinx_aws-vu9p-f1_shell-v04261818_201920_3 [connectivity]
   sp=vadd_1.in1:DDR[0]
   sp=vadd_1.in2:DDR[0]
   sp=vadd_1.out:DDR[0]

   2.1. The AWS F1 shell is targeted by the platform option. The string reflects the name of the most latest shell, which can be found on the aws-fpga repo. The platform should be pointed to the xpfm file. **platform=$AWS PLATFORM**, for example.

   2.2. The kernel arguments are connected to DDR[0], which is the DDR interface in the AWS F1 shell, using the sp options. On AWS F1, keeping the same settings as the U200 would result in a functioning design. However, the sp parameters are updated to utilise DDR[0] in order to build the exact identical configuration and target the DDR interface in the AWS F1 shell.

3. Only these changes are required to convert the project from U200 to F1.

4. The same commands that were used for U200 may be used to develop the project for AWS F1. To know more about the design and development flow of the accelerated application for the Alveo U200 using the ***Xilinx Vitis Unified Software platform***, refer to our ***White Paper***.

5. When targeting AWS F1, you'll need to take the extra step of building an Amazon FPGA Image (AFI). This is accomplished using AWS's create vitis afi.sh command. The steps and information on this are discussed in the above AFI section.

Run the App on AWS F1

1. To source the Vitis runtime environment, run the following command.

source $AWS_FPGA_REPO_DIR/vitis_runtime_setup.sh

2. Use the .awsxclbin FPGA binary to run the host application.

./host app_design.awsxclbin

3. The program's success will be shown by the messages below.

Found Platform

Platform Name: Xilinx

INFO: Reading ./vadd.awsxclbin

Loading: './vadd.awsxclbin'

TEST PASSED

Summary

Vitis AI is a dedicated development environment for boosting AI inference on Xilinx embedded platforms, Alveo accelerator cards, or cloud-based FPGA instances. **The Vitis development environment brings together Alveo™ technologies and Amazon EC2 F1 instances to create FPGA-accelerated apps.** The Vitis® flow is built on industry-standard programming languages for both software and hardware, as well as an open-source runtime library and optimization compiler technologies.

This method allows applications to move seamlessly between acceleration platforms. Xilinx was able to effortlessly convert over 40+ designs from the Alveo U200 platform to F1 instances using the Vitis tool flow, with no changes to the kernel source code and only minimal cosmetic changes to the application source code. ***Follow here*** to know more about the design flow and the complete

logic fruit Technologies

Thank You!

Does anyone have any questions?

Contact Us

Gurugram (Headquarter)

806, 8th Floor
BPTP Park Centra Sector–30,
NH–8 Gurgaon – 122001
Haryana (India)

info@logic-fruit.com

+91-124 4643950

Bengaluru (R&D House)

Sy. No 118, 3rd Floor,
Gayathri Lakefront,
Outer Ring Road, Hebbal,
Bangalore - 560 024

sales@logic-fruit.com

+91 80-69019700/01

United States (Sales Office)

Logic Fruit Technologies
INC 691 S Milpitas Blvd
Ste 217 (Room 9)
Milpitas CA 95035

info@logic-fruit.com

+1-408 338 9743