



Versal ACAP: Vitis Tool Flow

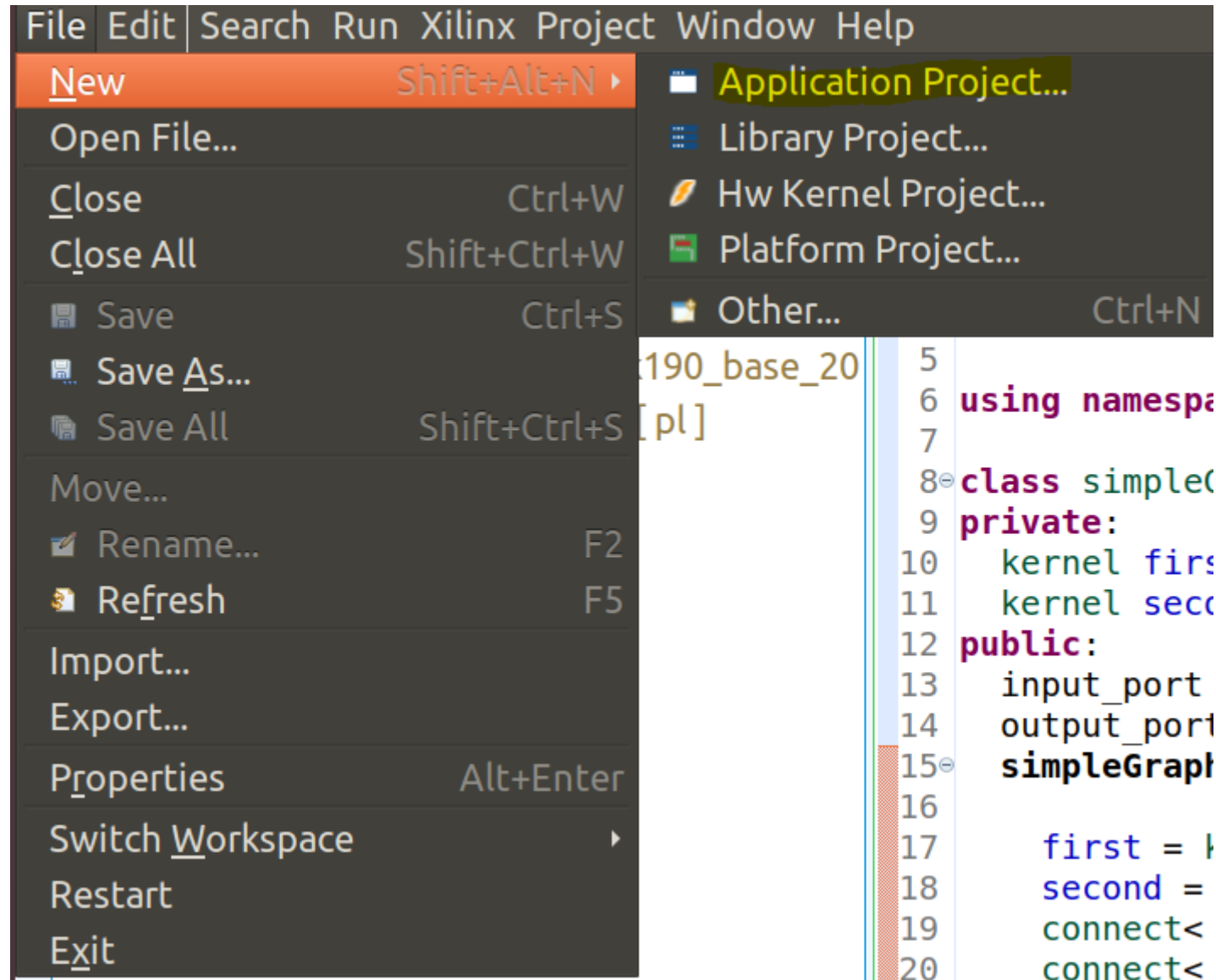
Revision History

Date	Version	Description
11/07/23	1.0	Initial version for flow introduction.

© Copyright 2021 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

NOTICE OF DISCLAIMER: The information disclosed to you hereunder (the "Information") is provided "AS-IS" with no warranty of any kind, express or implied. Xilinx does not assume any liability arising from your use of the Information. You are responsible for obtaining any rights you may require for your use of this Information. Xilinx reserves the right to make changes, at any time, to the Information without notice and at its sole discretion. Xilinx assumes no obligation to correct any errors contained in the Information or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE INFORMATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

Starting out with the AI Engine tools



Starting out with the AI Engine tools



New Application Project

Platform

Choose a platform for your project. You can also create an application from XSA through the 'Create a new platform from hardware (XSA)' tab.

Select a platform from repository

Create a new platform from hardware (XSA)

Find:

+Add

Manage

Name	Board	Flow	Vendor	Path
xilinx_vck190_base_20.xd		Embedded Accel	xilinx.cc	\$XILINX_VITIS/platforms/xilinx_vck190_base_202110_1/xilinx_vck190_base_202110_1.xpfm
xilinx_vck190_base_20.xd		Embedded Accel	xilinx.cc	\$XILINX_VITIS/platforms/xilinx_vck190_base_202120_1/xilinx_vck190_base_202120_1.xpfm

Platform Info

General Info

Name:

xilinx_vck190_base_202110_1

Part:

xcvc1902-vsva2197-2MP-e-S

Family:

versal

Description:

A base platform targeting VCK190 which is the first Versal AI Core series evaluation kit, enabling designers to develop solutions using AI and DSP engines capable of delivering over 100X greater compute performance compared to current server class CPUs. This board includes 8GB of DDR4

Acceleration Resources

Clock Frequencies

Clock	Frequency (MHz)
CPU	1
PL 0	150.000000
PL 1	100.000000
PL 2	300.000000
PL 3	75.000000
PL 4	200.000000
PL 5	400.000000
PL 6	600.000000

Domain Details

Domains

Domain name	Details
aiengine	CPU: ai_engine OS: aie_runtime
xrt	CPU: cortex-a72 OS: linux

?

< Back

Next >

Cancel

Finish

Starting out with the AI Engine tools



Different Vitis Version has different embedded platforms package

Vivado (HW Developer)	Vitis (SW Developer)	Vitis Embedded Platforms	Power Design Manager	Alveo Packages
-----------------------	----------------------	--------------------------	----------------------	----------------

Version

2023.2

2023.1

2022.2

2022.1

Vitis Embedded
Archive

Vitis Embedded Base Platforms - 2023.2

Important Information

The following base platforms are added to Vitis Installer. You can use them directly after installing Vitis.

- xilinx_zcu102_base_202320_1
- xilinx_zcu102_base_dfx_202320_1
- xilinx_zcu104_base_202320_1
- xilinx_vmk180_base_202320_1
- xilinx_vck190_base_202320_1
- xilinx_vck190_base_dfx_202320_1
- xilinx_vek280_es1_base_202320_1

📄 ZCU102 Base DFX 2021.1 (ZIP - 80.38 MB)

MD5 SUM Value : b07dafc2bb16a38ec5705bf9ff0d07b7

📄 ZCU104 Base 2021.1 (ZIP - 29.31 MB)

MD5 SUM Value : ff4d05ab2ad245eccd2c7216a29536f2

📄 ZC706 Base 201.1 (ZIP - 24.79 MB)

MD5 SUM Value : eb050f789639ab75345851dcc8f819b5

📄 VCK190 Base 2021.1 (ZIP - 24.93 MB)

MD5 SUM Value : a57b232c230669a54b718103c2cde12a

📄 VMK180 Base 2021.1 (ZIP - 23.41 MB)

MD5 SUM Value : d4f3d7c7cf01da399831bdacd67c3733

Starting out with the AI Engine tools



New Application Project

Application Project Details

Specify the application project name and its system project properties

Application project name: Test

System Project

Create a new system project for the application or select an existing one from the workspace

Select a system project

KernelProg_Scalar_system

multi_kernel_system

simple_test_system

single_node_graph_system

+ Create new...

System project details

System project name: Test_system

Target processor

Select target processor for the Application project.

Processor	Associated applications
ai_engine	Test
psv_cortexa72 SMP	

Show all processors in the hardware specification

?

< Back

Next >

Cancel

Finish

Starting out with the AI Engine tools



New Application Project

Domain

Select a domain for your project or create a new domain

Select the domain that the application would link to or create a new domain

Note: New domain created by this wizard will have all the requirements of the application template selected in the next step

Select a domain

aiengine

Domain details

Name:

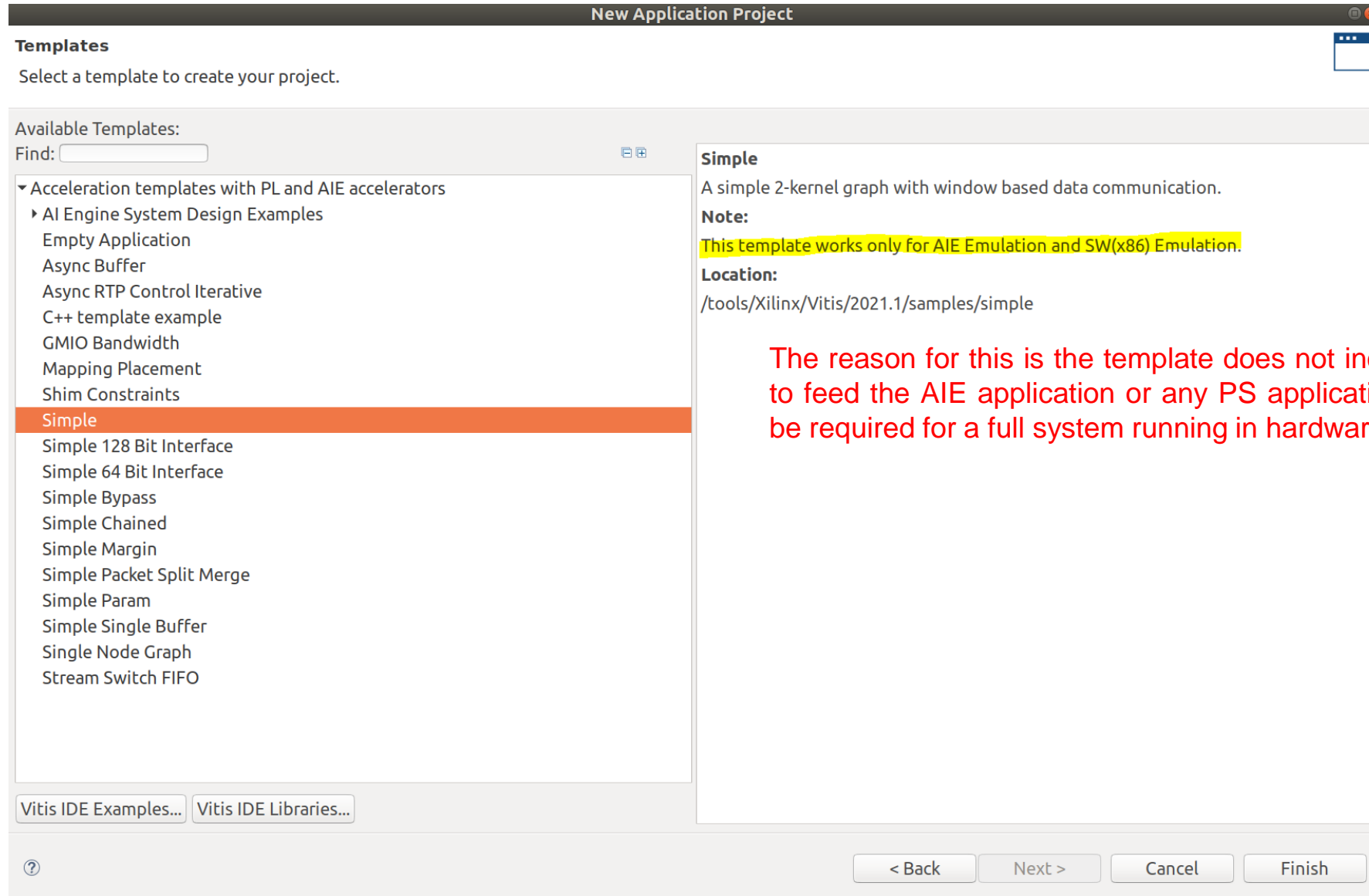
Display Name:

Operating System:

Processor:

? < Back Next > Cancel Finish

Starting out with the AI Engine tools



Starting out with the AI Engine tools



▼ simple_test_system [xilinx_vck190_base_202110_1]

▶ simple_test_system_hw_link [pl]

▼ simple_test [aiengine]

▶ Trace Compass

▶ Binaries

▶ Archives

▶ Includes

▼ data

golden.txt

input.txt

Input Data and Ground Truth.

▶ Emulation-AIE

▼ src

▼ kernels

include.h

kernels.cc

AIE Data Flow Graph Definition.

kernels.h

project.cpp

project.h

Feed Data and Run whole AIE execution.

▶ _ide

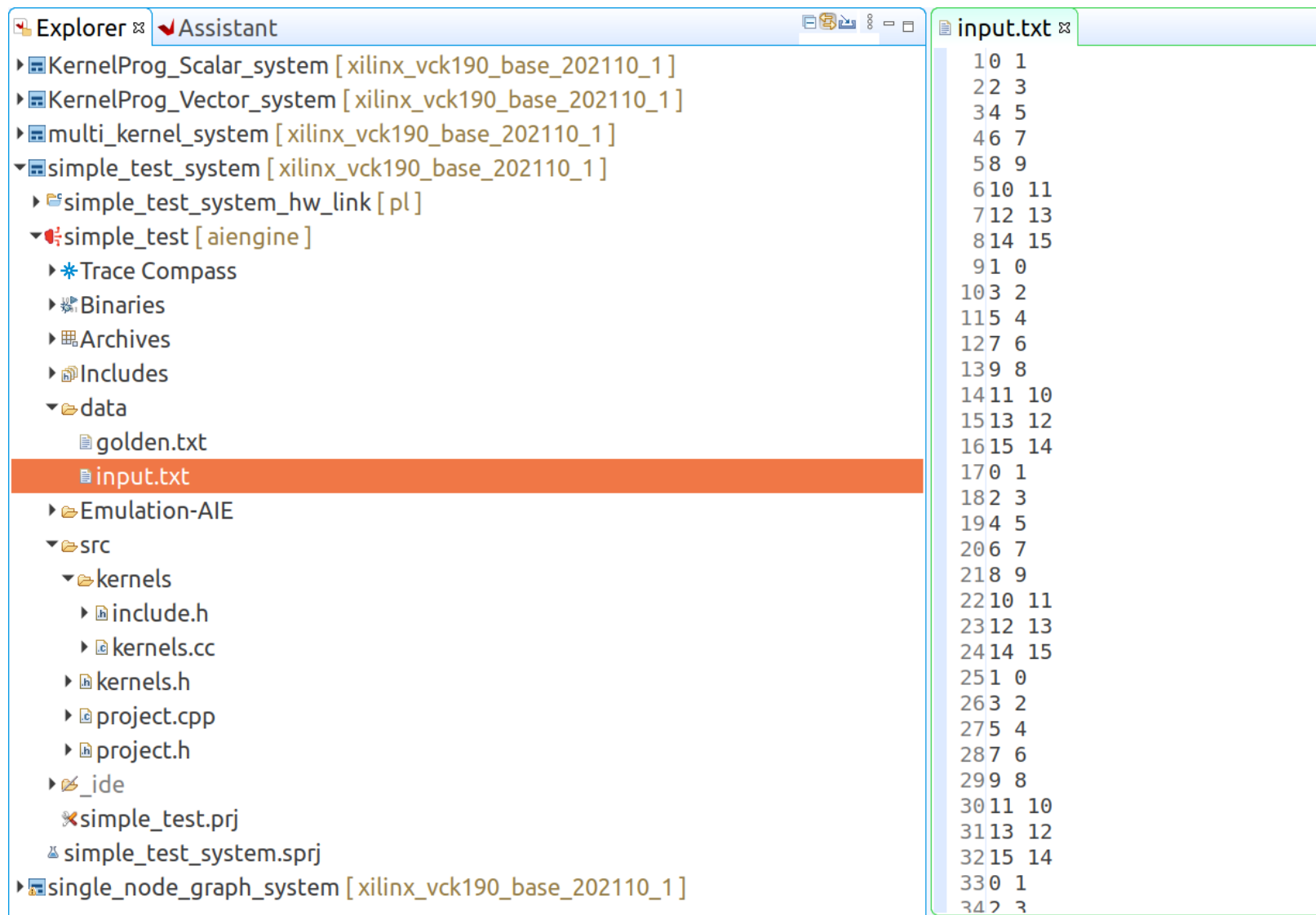
simple_test.prj

simple_test_system.sprj

Starting out with the AI Engine tools



Input Data



Starting out with the AI Engine tools



Ground Truth

The screenshot shows the Xilinx Vitis IDE interface. The Explorer tab on the left displays the project structure, including folders like 'data', 'golden.txt', 'input.txt', 'Emulation-AIE', 'src', 'kernels', and 'project.cpp'. The Assistant tab on the right shows a list of files, with 'golden.txt' selected. The content of 'golden.txt' is displayed on the right, showing a list of numbers.

File	Content
golden.txt	1 0 2
golden.txt	2 4 6
golden.txt	3 8 10
golden.txt	4 12 14
golden.txt	5 16 18
golden.txt	6 20 22
golden.txt	7 24 26
golden.txt	8 28 30
golden.txt	9 2 0
golden.txt	10 6 4
golden.txt	11 10 8
golden.txt	12 14 12
golden.txt	13 18 16
golden.txt	14 22 20
golden.txt	15 26 24
golden.txt	16 30 28
golden.txt	17 0 2
golden.txt	18 4 6
golden.txt	19 8 10
golden.txt	20 12 14
golden.txt	21 16 18
golden.txt	22 20 22
golden.txt	23 24 26
golden.txt	24 28 30
golden.txt	25 2 0
golden.txt	26 6 4
golden.txt	27 10 8
golden.txt	28 14 12
golden.txt	29 18 16
golden.txt	30 22 20
golden.txt	31 26 24
golden.txt	32 30 28
golden.txt	33 0 2
golden.txt	34 4 6

Starting out with the AI Engine tools



Text Comparison

Text Compare			
simple_test/data/golden.txt		simple_test/data/input.txt	
1 0 2		1 0 1	
2 4 6		2 2 3	
3 8 10		3 4 5	
4 12 14		4 6 7	
5 16 18		5 8 9	
6 20 22		6 10 11	
7 24 26		7 12 13	
8 28 30		8 14 15	
9 2 0		9 1 0	
10 6 4		10 3 2	
11 10 8		11 5 4	
12 14 12		12 7 6	
13 18 16		13 9 8	
14 22 20		14 11 10	
15 26 24		15 13 12	
16 30 28		16 15 14	
17 0 2		17 0 1	
18 4 6		18 2 3	
19 8 10		19 4 5	
20 12 14		20 6 7	
21 16 18		21 8 9	
22 20 22		22 10 11	
23 24 26		23 12 13	
24 28 30		24 14 15	
25 2 0		25 1 0	
26 6 4		26 3 2	
27 10 8		27 5 4	
28 14 12		28 7 6	
29 18 16		29 9 8	
30 22 20		30 11 10	
31 26 24		31 13 12	

Ground Truth

Input Data

Starting out with the AI Engine tools



AIE Data Flow Graph Definition

The screenshot displays the Xilinx Vitis IDE interface. On the left, the Explorer pane shows a project structure for 'simple_test_system'. The 'src' directory is expanded, showing a 'kernels' subdirectory. The main editor area is split into two panes. The left pane shows 'include.h' with the following content:

```
1 // 67d7842dbbe25473c3c32b93c0da8047785f30d78
2
3 #ifndef FUNCTION_INCLUDES_H
4 #define FUNCTION_INCLUDES_H
5
6 #define NUM_SAMPLES 32
7
8 #endif
9
```

The right pane shows 'kernels.cc' with the following content:

```
1 /* A simple kernel
2 */
3 #include <adf.h>
4 #include "include.h"
5
6
7 void simple(input_window_cint16 * in, output_window_cint16 * out) {
8     cint16 c1, c2;
9     for (unsigned i=0; i<NUM_SAMPLES; i++) {
10         window_readincr(in, c1);
11         c2.real = c1.real+c1.imag;
12         c2.imag = c1.real;
13
14         window_writeincr(out, c2);
15     }
16 }
17
```

Starting out with the AI Engine tools



AIE Data Flow Graph Definition

kernels.cc

```
1 /* A simple kernel
2 */
3 #include <adf.h>
4 #include "include.h"
5
6
7 void simple(input_window_cint16 * in, output_window_cint16 * out) {
8     cint16 c1, c2;
9     for (unsigned i=0; i<NUM_SAMPLES; i++) {
10         window_readincr(in, c1);
11         c2.real = c1.real+c1.imag;
12         c2.imag = c1.real-c1.imag;
13
14         window_writeincr(out, c2);
15     }
16 }
17
```

Kernel Function

Each AIE support one and multiple Kernel, but one Kernel only run on one of AIE.

Different Kernel can source multiple algorithm to define own function.

Starting out with the AI Engine tools



AIE Data Flow Graph Definition

Table 34: Data Type and PLIO Width

Data Type	PLIO 32 bit	PLIO 64 bit	PLIO 128 bit
	<code>adf::input_plio in = adf::input_plio::create('D ataIn1', adf::plio_32_bits, 'input.csv');</code>	<code>adf::input_plio in = adf::input_plio::create('D ataIn1', adf::plio_64_bits, 'input.csv');</code>	<code>adf::input_plio in = adf::input_plio::create('D ataIn1', adf::plio_128_bits, 'input.csv');</code>
int8	Four data columns expected. For example: CMD, D,D,D,D,TKEEP,TLAST DATA, 6,8,3,2,-1,0	Eight data columns expected. For example: CMD, D,D,D,D,D,D,D,D, TKEEP,TLAST DATA, 6,8,3,2,6,8,3,2, -1,0	16 data columns expected. For example: CMD, D,D,D,D,D,D,D,D,D,D,D,D,D,D,D,D, D, TKEEP,TLAST DATA, 6,8,3,2,6,8,3,2,6,8,3,2,6,8,3, 2, -1,0
int16	Two data columns expected. For example: CMD,D,D, TKEEP,TLAST DATA, 24,18, -1,0	Four data columns expected. For example: CMD, D,D,D,D, TKEEP,TLAST DATA, 24,18,24,18, -1,0	Eight data columns expected. For example: CMD,D,D,D,D,D,D,D,D,D,TKEEP,TLAST DATA,24,18,24,18,24,18,24,18,- 1,0
int32	One data column expected. CMD, D, TKEEP, TLAST DATA, 2386, -1, 0	Two data columns expected. For example: CMD, D, D, TKEEP, TLAST DATA, 2386, 2386, -1, 0	Four data columns expected. For example: CMD,D,D,D,D,TKEEP,TLAST DATA,2386,2386,2386,2386,-1,0
int64	N/A	One data column expected. CMD, D, TKEEP, TLAST DATA, 45678, -1, 0	Two data columns expected. For example: CMD, D, D, TKEEP, TLAST DATA, 45678, 95578, -1, 0
cint16	Two data columns expected to represent one <code>cint</code> value (real, imag). For example: CMD, D,D,TKEEP, TLAST DATA,1980,485,-1,0	Two <code>cint</code> values (real, imag) represented by four data columns expected. For example: CMD, D,D,D,D, TKEEP, TLAST DATA, 1980, 45, 180, 85, -1, 0	Four <code>cint</code> values (real, imag) represented by 8 data columns expected. For example: CMD, D,D,D,D,D,D,D,D, TKEEP, TLAST DATA, 1980,485,180,85,980,48,190,45, -1,0
cint32	N/A	One <code>cint</code> value (real, imag) represented by two data columns expected. For example: CMD, D, D, TKEEP, TLAST DATA, 1980, 485, -1, 0	Two <code>cint</code> values represented by four data columns expected. For example: CMD, D, D, D, D, TKEEP, TLAST DATA, 1980, 45, 180, 85, -1, 0

CINT16 means 16-bit Real number with 16-bit Imaginary number

```
input.txt
10 1
22 3
34 5
46 7
58 9
610 11
712 13
814 15
91 0
103 2
115 4
127 6
139 8
1411 10
1513 12
```

Starting out with the AI Engine tools



AIE Data Flow Graph Definition

kernels.cc

```
1 /* A simple kernel
2 */
3 #include <adf.h>
4 #include "include.h"
5
6
7 void simple(input_window_cint16 * in, output_window_cint16 * out) {
8     cint16 c1, c2;
9     for (unsigned i=0; i<NUM_SAMPLES; i++) {
10         window_readincr(in, c1);
11         c2.real = c1.real+c1.imag;
12         c2.imag = c1.real-c1.imag;
13
14         window_writeincr(out, c2);
15     }
16 }
17
```

32x32-bit/8 = 128bytes

Starting out with the AI Engine tools



Feed Data and Run whole AIE execution

The screenshot displays the Xilinx Vitis IDE interface. On the left, the 'Explorer' pane shows a project tree with the following structure:

- KernelProg_Scalar_system [xilinx_vck190_base_202110_1]
- KernelProg_Vector_system [xilinx_vck190_base_202110_1]
- multi_kernel_system [xilinx_vck190_base_202110_1]
- simple_test_system [xilinx_vck190_base_202110_1]
 - simple_test_system_hw_link [pl]
 - simple_test [aiengine]
 - Trace Compass
 - Binaries
 - Archives
 - Includes
 - data
 - golden.txt
 - input.txt
 - Emulation-AIE
 - src
 - kernels
 - include.h
 - kernels.cc
 - kernels.h
 - project.cpp
 - project.h
 - _ide
 - simple_test.prj
 - simple_test_system.sprj

The 'project.h' file is selected in the Explorer. On the right, the 'project.cpp' file is open, showing the following code:

```
1 // 67d7842dbbe25473c3c32b93c0da8047785f30d78e8a024de1b57352245f9689
2
3 #include <adf.h>
4 #include "kernels.h"
5
6 using namespace adf;
7
8 class simpleGraph : public adf::graph {
9 private:
10     kernel first;
11     kernel second;
12 public:
13     input_port in;
14     output_port out;
15     simpleGraph(){
16
17         first = kernel::create(simple);
18         second = kernel::create(simple);
19         connect< window<128> > net0 (in, first.in[0]);
20         connect< window<128> > net1 (first.out[0], second.in[0]);
21         connect< window<128> > net2 (second.out[0], out);
22
23         source(first) = "kernels/kernels.cc";
24         source(second) = "kernels/kernels.cc";
25
26         runtime<ratio>(first) = 0.1;
27         runtime<ratio>(second) = 0.1;
28     }
29 };
30
31
```

Starting out with the AI Engine tools

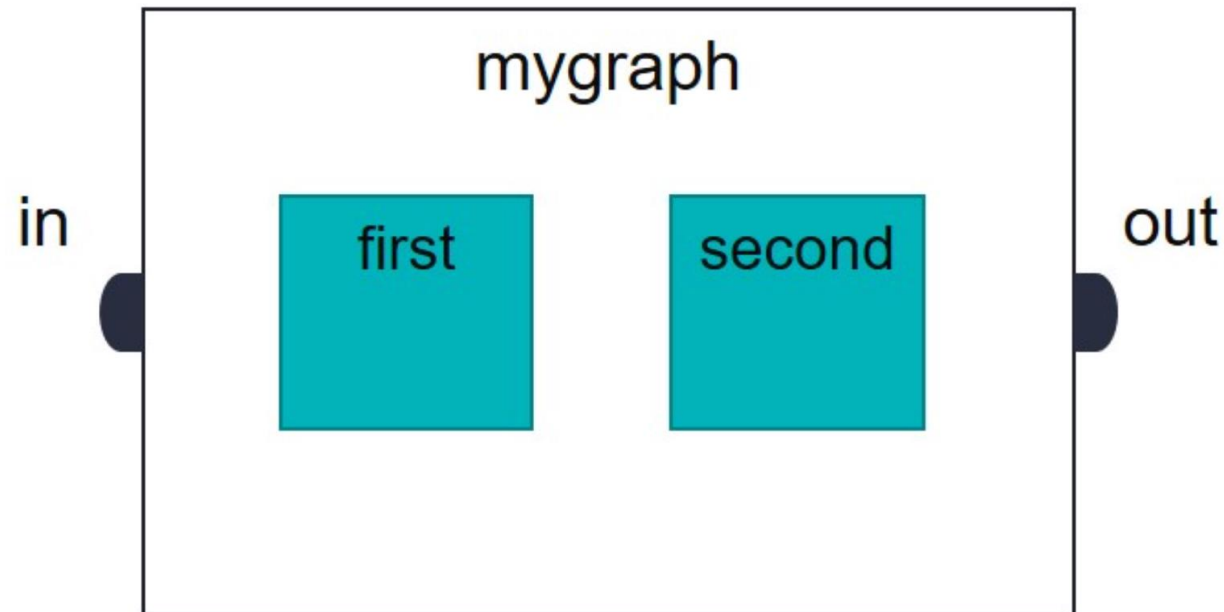


Feed Data and Run whole AIE execution

```
class simpleGraph : public adf::graph {  
private:  
    kernel first;  
    kernel second;  
public:  
    input_port in;  
    output_port out;
```

→ Define two kernel

→ Define one input and one output from PL

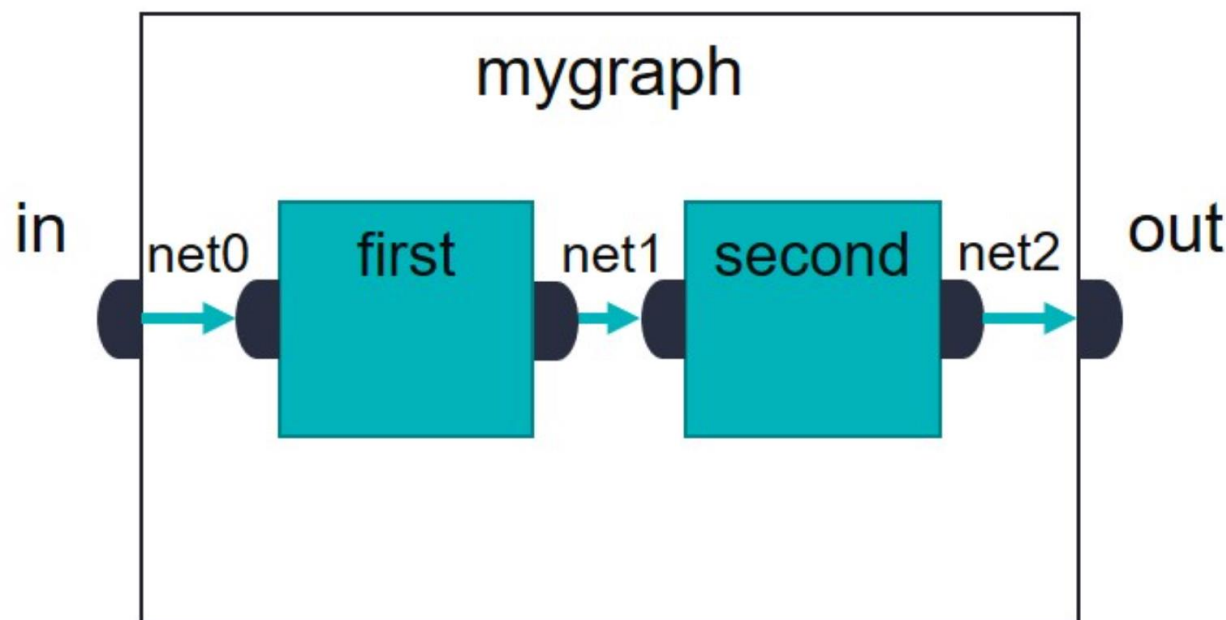


Starting out with the AI Engine tools



Feed Data and Run whole AIE execution

```
first = kernel::create(simple);  
second = kernel::create(simple);  
connect< window<128> > net0 (in, first.in[0]);  
connect< window<128> > net1 (first.out[0], second.in[0]);  
connect< window<128> > net2 (second.out[0], out);
```



Starting out with the AI Engine tools



Feed Data and Run whole AIE execution

```
source(first) = "kernels/kernels.cc";  
source(second) = "kernels/kernels.cc";
```

Define Kernel function

```
runtime<ratio>(first) = 0.1;  
runtime<ratio>(second) = 0.1;
```

Both kernels are estimated to run at 10% of the processing time of a single AI Engine

Starting out with the AI Engine tools



Feed Data and Run whole AIE execution

The screenshot displays the Xilinx Vitis IDE interface. On the left, the 'Explorer' pane shows a project tree with the following structure:

- KernelProg_Scalar_system [xilinx_vck190_base_202110_1]
- KernelProg_Vector_system [xilinx_vck190_base_202110_1]
- multi_kernel_system [xilinx_vck190_base_202110_1]
- simple_test_system [xilinx_vck190_base_202110_1]
 - simple_test_system_hw_link [pl]
 - simple_test [aiengine]
 - Trace Compass
 - Binaries
 - Archives
 - Includes
 - data
 - golden.txt
 - input.txt
 - Emulation-AIE
 - src
 - kernels
 - include.h
 - kernels.cc
 - kernels.h
 - project.cpp (highlighted)
 - project.h
 - _ide
 - simple_test.prj
 - simple_test_system.sprj

On the right, the 'project.cpp' file is open, showing the following code:

```
1 // 67d7842dbbe25473c3c32b93c0da8047785f30d78e8a024de1b57352245f9689
2
3 #include <adf.h>
4 #include "kernels.h"
5 #include "project.h"
6
7 using namespace adf;
8
9 simpleGraph mygraph;
10 simulation::platform<1,1> platform("data/input.txt", "data/output.txt");
11 connect<> net0(platform.src[0], mygraph.in);
12 connect<> net1(mygraph.out, platform.sink[0]);
13
14 int main(void) {
15     mygraph.init();
16     mygraph.run(4);
17     mygraph.end();
18     return 0;
19 }
20
```

Annotations in the image highlight specific parts of the code:

- Line 10: `simulation::platform<1,1> platform("data/input.txt", "data/output.txt");` is highlighted with a red box, with the text "Define Real Data input and output file" to its right.
- Line 16: `mygraph.run(4);` is highlighted with a red box, with the text "Run for 4 iteration" to its right.

Starting out with the AI Engine tools



Build and Run Project

Explorer

Assistant

- KernelProg_Scalar_system [xilinx_vck190_base_202110_1]
- KernelProg_Vector_system [xilinx_vck190_base_202110_1]
- multi_kernel_system [xilinx_vck190_base_202110_1]
- simple_test_system [xilinx_vck190_base_202110_1]
 - simple_test_system_hw_link [pl]
 - simple_test [aieengine]
 - Trace Compass
 - Binaries
 - Archives
 - Includes
 - data
 - golden.txt
 - input.txt
 - Emulation-AIE
 - src
 - kernels
 - include.h
 - kernels.cc
 - kernels.h
 - project.cpp
 - project.h
 - _ide
- simple_test.prj
- simple_test_system.sprj
- single_node_graph_system [xilinx_vck190_base_202110_1]

project.h project.cpp simple_test

Application Project Settings

Active build configuration: Emulation-AIE

General

Project name: simple_test

Platform: xilinx_vck190_base_202110_1

Runtime: C/C++

Domain: aieengine

CPU: ai_engine

OS: aie_runtime

AIE Compiler Options

Target: AIE Simulation

Top-Level File: "\${workspace_loc}/\${ProjName}/src/project.cpp"

- **Emulation-SW:**
Compiles for the X86 processor. This is used for **functional simulation only** (not cycle approximate).
- **Emulation-AIE:**
Compiles for the AI Engine. The code will be simulated using a System-C model for the AI Engine. This **provides a cycle approximate and functional simulation**.
- **Hardware:**
Compiles for the Hardware target.

Starting out with the AI Engine tools



Build and Run Project

▼ simple_test_system [xilinx_vck190_base_202110_1]
 ▶ simple_test_system_hw_link [pl]
 ▼ simple_test [aiengine]
 ▶ Trace Compass
 ▶ Binaries
 ▶ Archives
 ▶ Includes
 ▶ data
 ▼ Emulation-AIE
 ▼ aiesimulator_output
 ▶ Work
 ▶ libadf.a
 AIECompiler.log
 aiesimulator.log
 build.log
 diag_report.log
 foo.vcd
 makefile
 pl_sample_counts
 run.log
 simple_test_Emulation-AIE.build.ui.log
 xcd.log
 xsc_report.log
 ▶ src
 ▶ _ide
 simple_test.prj
 simple_test_system.sprj

▼ Emulation-AIE
 ▼ aiesimulator_output
 ▼ data
 output.txt
 aiesim_options.txt
 default.aierun_summary
 vitis_analyzer_8181.backup.jou
 vitis_analyzer_8181.backup.log
 vitis_analyzer.jou
 vitis_analyzer.log

▼ Emulation-AIE
 ▶ aiesimulator_output
 ▼ Work
 ▶ aie
 ▶ arch
 ▶ config
 ▶ noc
 ▶ ps
 ▶ reports
 ▶ scripts
 ▶ temp
 project.aiecompile_summary

Starting out with the AI Engine tools



Verify Output and check AIE execution status

▼ simple_test [aiengine]

▶ Trace Compass

▶ Binaries

▶ Archives

▶ Includes

▼ data

golden.txt

input.txt

▼ Emulation-AIE

▼ aiesimulator_output

▼ data

output.txt

aiesim_options.txt

default.aierun_summary

vitis_analyzer_8181.backup.jou

vitis_analyzer_8181.backup.log

vitis_analyzer.jou

vitis_analyzer.log

Runtime: C/C++

Domain: aiengine ...

CPU: ai_engine

OS: aie_runtime

New

Open

Copy Ctrl+C

Paste Ctrl+V

Delete

Refresh F5

Move...

Rename... F2

Compare With

Replace With

Team

Each Other After Transformation

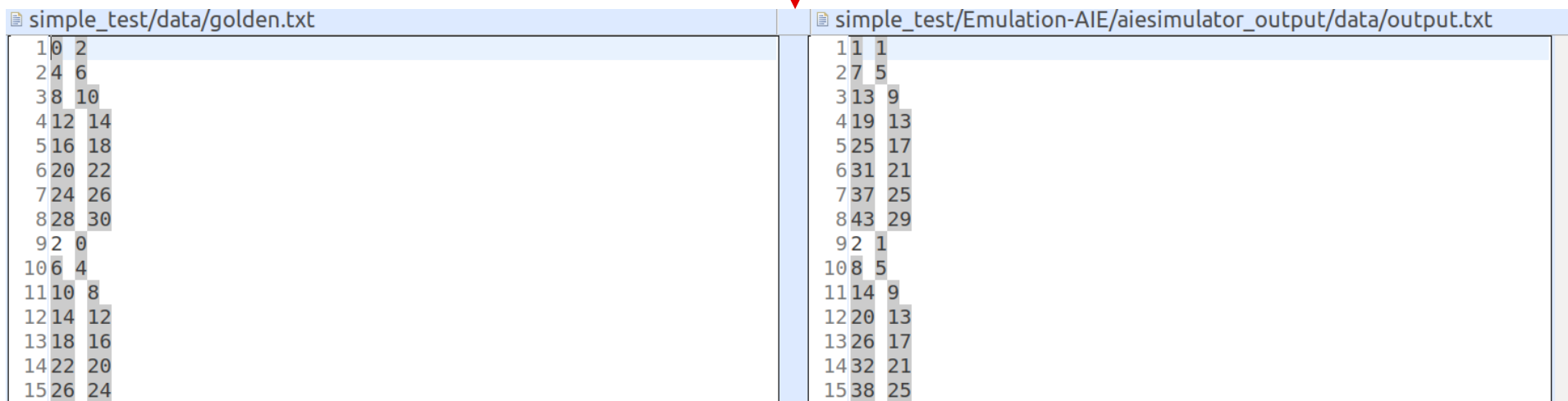
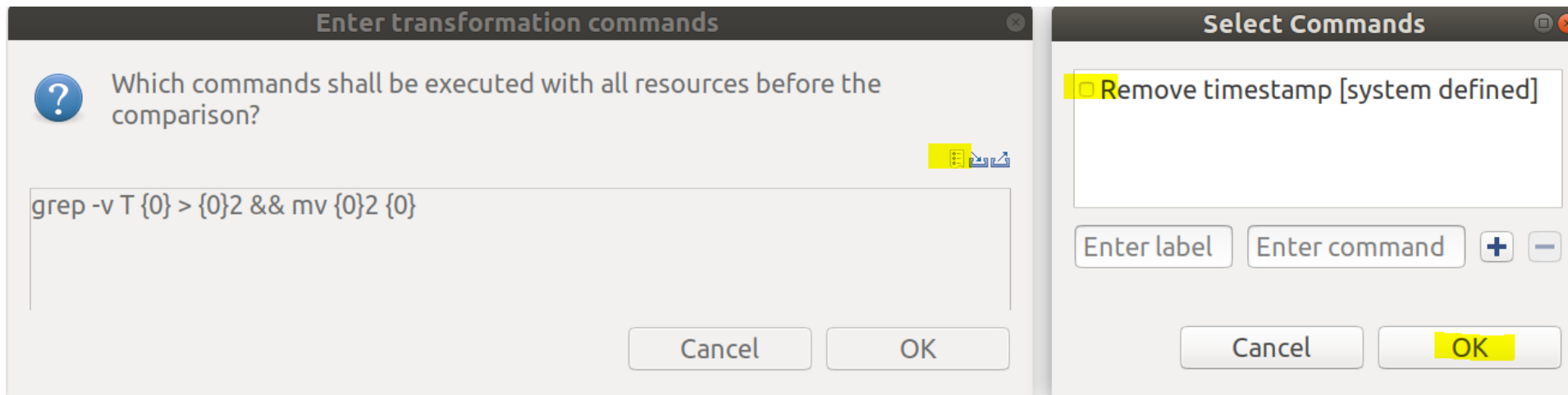
Each Other

Local History...

Starting out with the AI Engine tools



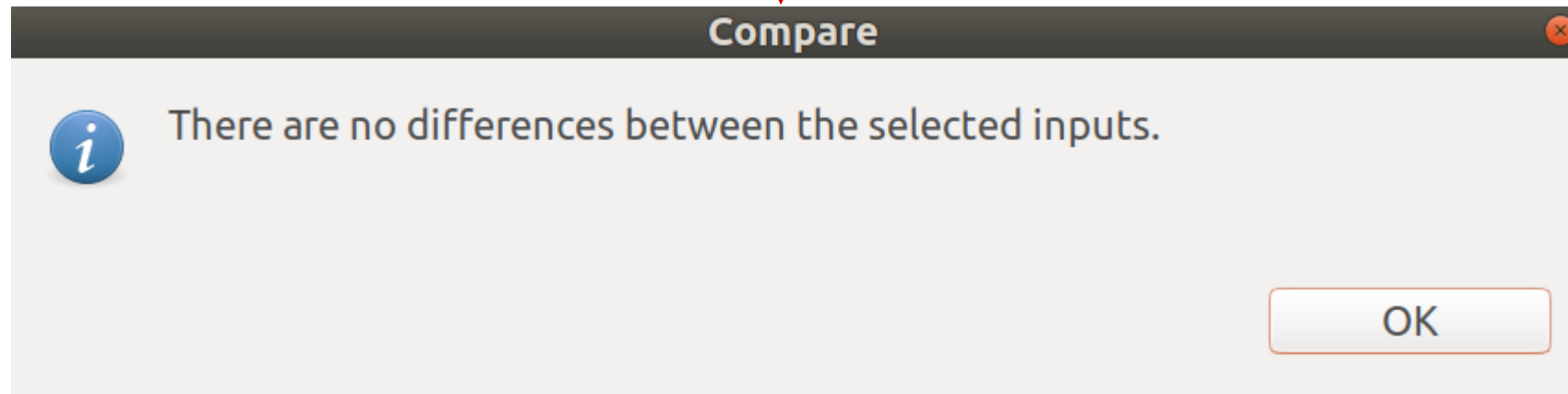
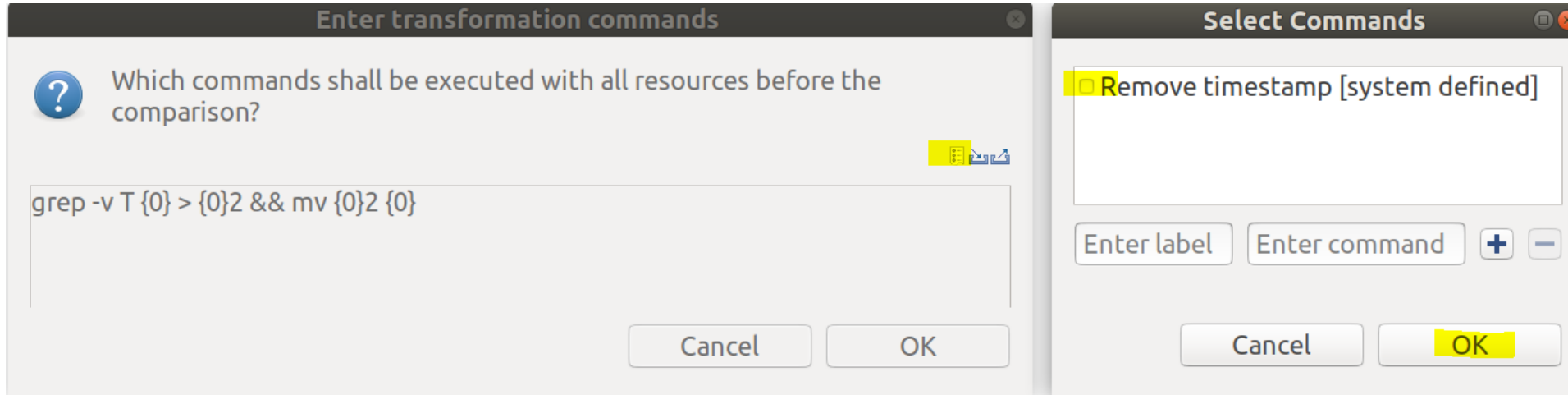
Verify Output and check AIE execution status



Starting out with the AI Engine tools



Verify Output and check AIE execution status



Starting out with the AI Engine tools



Verify Output and check AIE execution status

- ▼ Emulation-AIE
 - ▶ aiesimulator_output
 - ▼ Work
 - ▶ aie
 - ▶ arch
 - ▶ config
 - ▶ noc
 - ▶ ps
 - ▶ reports
 - ▶ scripts
 - ▶ temp

project.aiecompile_summary

Double Click

Starting out with the AI Engine tools



Vitis Analyzer

The screenshot displays the Vitis Analyzer web interface. The top navigation bar includes 'File', 'Tools', 'View', 'Layout', and 'Help', along with a search bar and a 'Default Layout' dropdown. The left sidebar shows a tree view for the 'project (...ngine Hardware)' with sub-items: Summary (highlighted with a red arrow), Kernel Guidance, Graph, Array, Constraints, Mapping Analysis, DMA Analysis, Lock Allocation, Log, and Core Compilation. The main content area shows the 'Summary' page for the 'project'. It includes a 'Summary' tab, a file path, and a large green 'Completed' status message with a 'Log' link. Below this, a table shows the project's version, start and completion times, elapsed time, platform, kernels, and core compilation status. The command line is also displayed at the bottom.

STATUS	Completed		
VERSION	Vitis AI Engine Compiler Release 2021.1. SW Build 3247384 on Thu Jun 10 19:36:07 MDT 2021		
STARTED	November 05, 2023 21:13:27	COMPLETED	November 05, 2023 21:14:07
ELAPSED	40s		
PLATFORM	VC1902		
KERNELS	Graph		
CORE COMPILATION	1 of 400 (0.25 %) Used Report		
Completed 1 of 1			
COMMAND LINE			

```
/tools/Xilinx/Vitis/2021.1/aietools/bin/unwrapped/linux64.o/aiecompiler
-v
--xlopt=0
-Xchess=main:darts.xargs=-nb
-include=/tools/Xilinx/Vitis/2021.1/aietools/include
-include=/tools/Xilinx/Vitis_HLS/2021.1/include
-include=../src
-include=../data
-include=../src/kernels
-target=hw
```

Starting out with the AI Engine tools



Vitis Analyzer

project (AI Engine Hardware) x

Summary x Graph x

Subgraph View
Subgraph View
Tile View
Flat View

Input: plio:platform.src[0] (data/input.tx)

src[0]

Stream

buf0
buf0d

Memory

in

first
simple

out

Memory

buf1

Memory

in

second
simple

out

Memory

buf2
buf2d

Stream

sink[0]

Output: plio:platform.sink[0] (data/output.tx)

mygraph

Graph Instance ID Kernel Runs on Source Column Row Schedule Runtime Ratio Graph Source

mygraph

first	i0	simple	AI Engine	kernels.cc	25	0	0	0.100	project.h:17:13
second	i1	simple	AI Engine	kernels.cc	25	0	1	0.100	project.h:18:14

Kernels PL Buffers Ports Nets Tiles

Starting out with the AI Engine tools



Vitis Analyzer

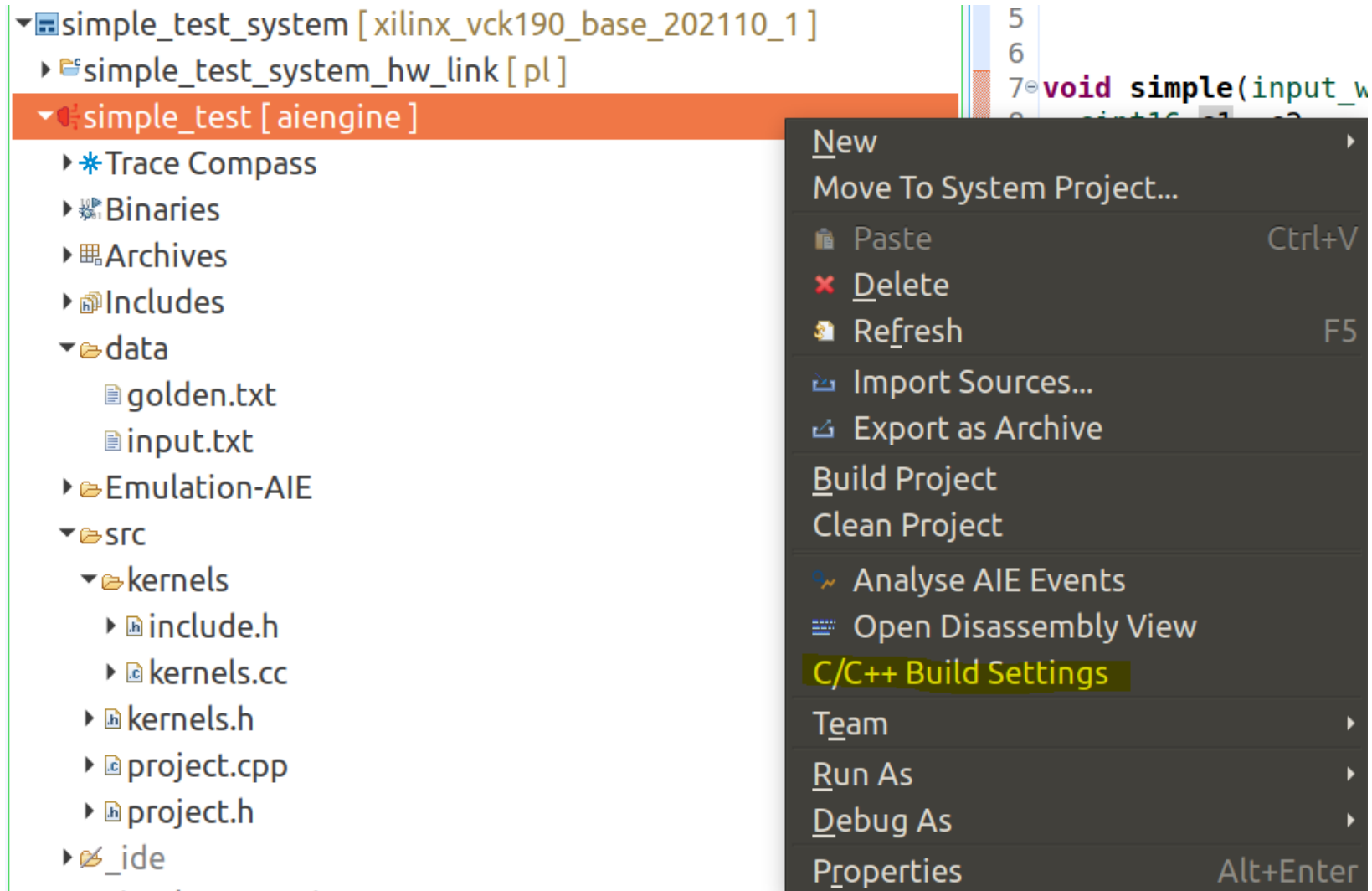
The Vitis Analyzer interface displays a project graph in Tile View. The graph shows two 'simple' kernels, 'first' and 'second', connected by data paths. Buffers (buf0, buf1, buf2) are used for data storage. The interface includes a left sidebar with navigation options like Summary, Graph, Array, Constraints, etc. A bottom table lists graph instances with details like ID, Kernel, and Runtime Ratio.

Graph Instance	ID	Kernel	Runs on	Source	Column	Row	Schedule	Runtime Ratio	Graph Source
mygraph									
first	i0	simple	AI Engine	kernels.cc	25	0	0	0.100	project.h:17:13
second	i1	simple	AI Engine	kernels.cc	25	0	1	0.100	project.h:18:14

Starting out with the AI Engine tools



Verify Output and check AIE execution status



Starting out with the AI Engine tools



Verify Output and check AIE execution status

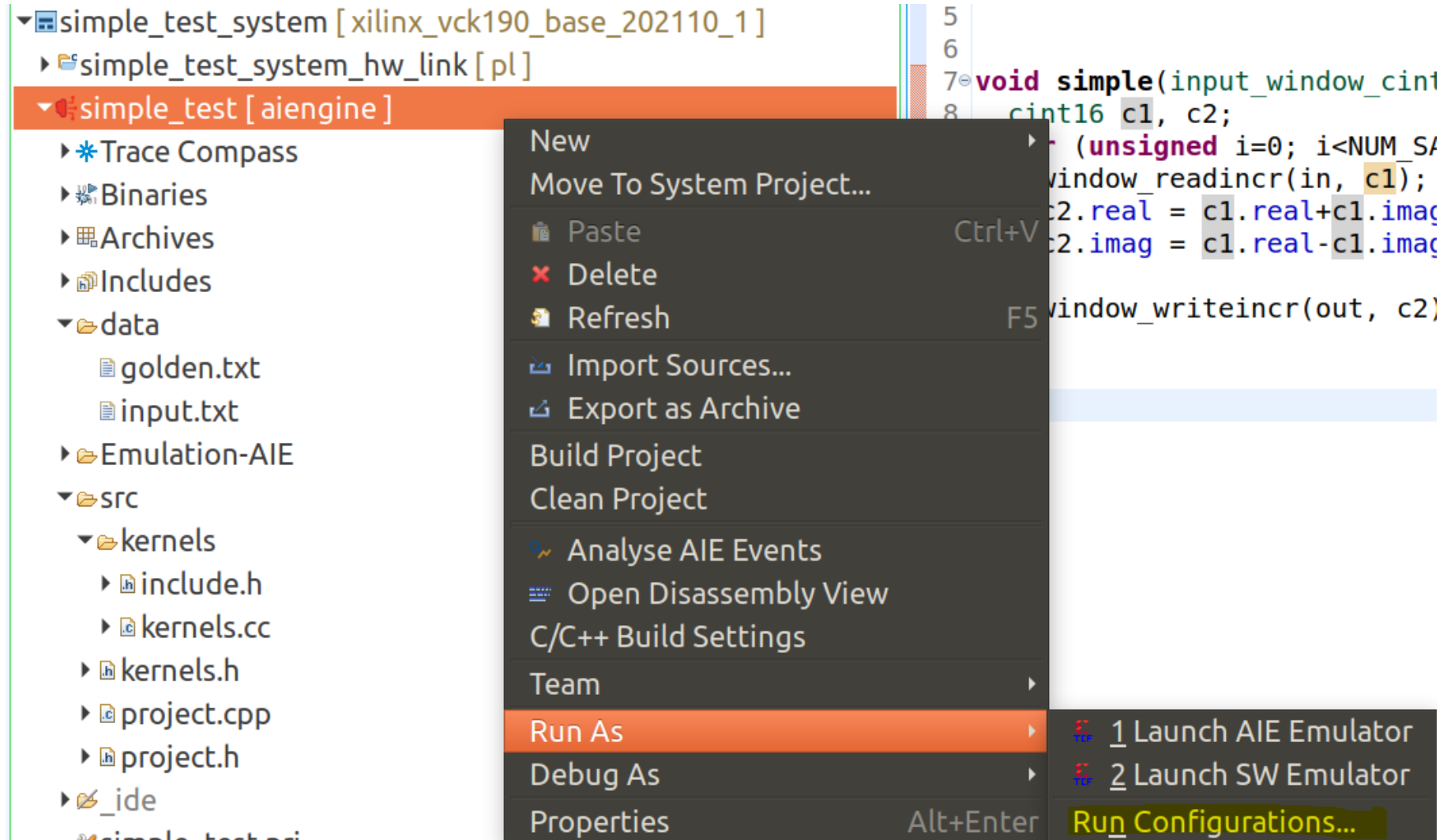
Optimization Options

- **xlopt:**
Enable kernel optimizations based on the specified value:
- 0: No kernel optimizations.
- 1: Computation of heap size, generation of guidance based on LLVM IR analysis, and insertion of loop pragmas.
- 2: The same optimizations as 1 with the addition of loop peeling for unrolled loops, and automatic inlining.

Starting out with the AI Engine tools



Verify Output and check AIE execution status



Starting out with the AI Engine tools



Verify Output and check AIE execution status – rerun the project

The screenshot shows the Xilinx Vitis IDE interface. On the left, a project tree lists several AI Engine Emulator instances, with 'AI Engine Emulator for simple_test Emulation-AIE' selected. The main panel displays the configuration for this emulator. The 'Name' field is 'AI Engine Emulator for simple_test Emulation-AIE'. The 'Debug Type' is 'Standalone Application Debug'. The 'Project' is 'simple_test'. The 'Build Configuration' is 'Emulation-AIE'. The 'Generate Trace' checkbox is checked. Under 'Generate Trace', the 'VCD' radio button is selected, and the 'VCD FileName' is 'foo'. The 'Generate Profile' checkbox is unchecked. Under 'Generate Profile', the 'Generate Function and Instruction repo' checkbox is checked, and the 'Generate all reports for selected Active' checkbox is checked. The '25_0' checkbox is also checked.

Value Change Dump (VCD)

1. time stamps
2. different event types
3. data associated with each event

These events describe the execution of the AI Engine application

Starting out with the AI Engine tools



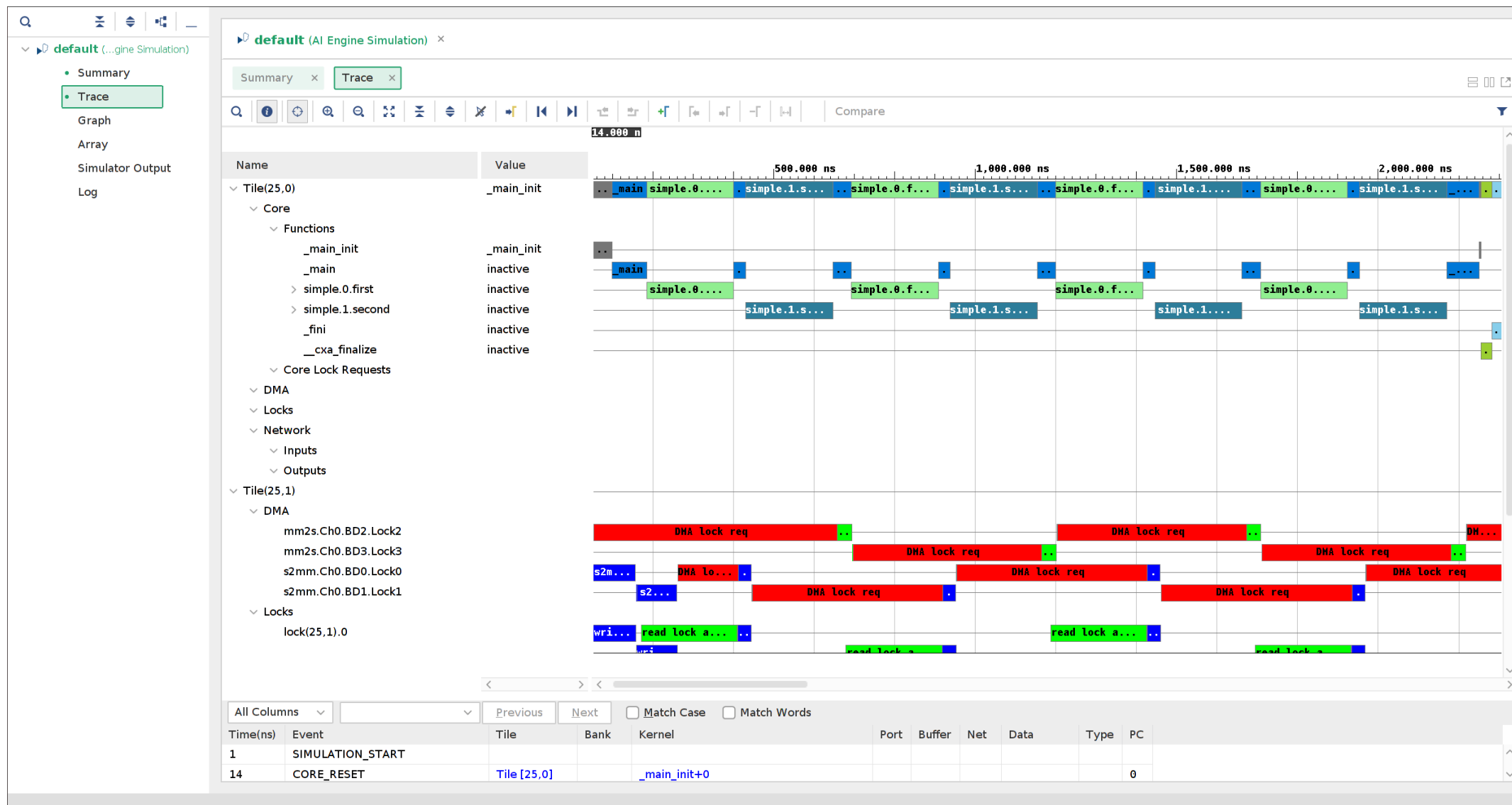
Verify Output and check AIE execution status

```
simple_test_system [ xilinx_vck190_base_202110_1 ]
├── simple_test_system_hw_link [ pl ]
├── simple_test [ aiengine ]
│   ├── Trace Compass
│   ├── Binaries
│   ├── Archives
│   ├── Includes
│   └── data
│       ├── golden.txt
│       ├── input.txt
│       └── Emulation-AIE
│           ├── aiesimulator_output
│           │   ├── data
│           │   │   ├── aiesim_options.txt
│           │   │   ├── default.aierun_summary Double Click
│           │   │   ├── vitis_analyzer_8181.backup.jou
│           │   │   ├── vitis_analyzer_8181.backup.log
│           │   │   ├── vitis_analyzer.jou
│           │   │   └── vitis_analyzer.log
│           ├── Work
│           └── libadf.a
│               ├── AIECompiler.log
│               ├── aiesimulator.log
│               ├── build.log
│               ├── diag_report.log
│               ├── foo.vcd
│               ├── makefile
│               ├── pl_sample_counts
│               ├── run.log
│               ├── simple_test_Emulation-AIE.build.ui.log
│               ├── xcd.log
│               └── xsc_report.log
```

Starting out with the AI Engine tools



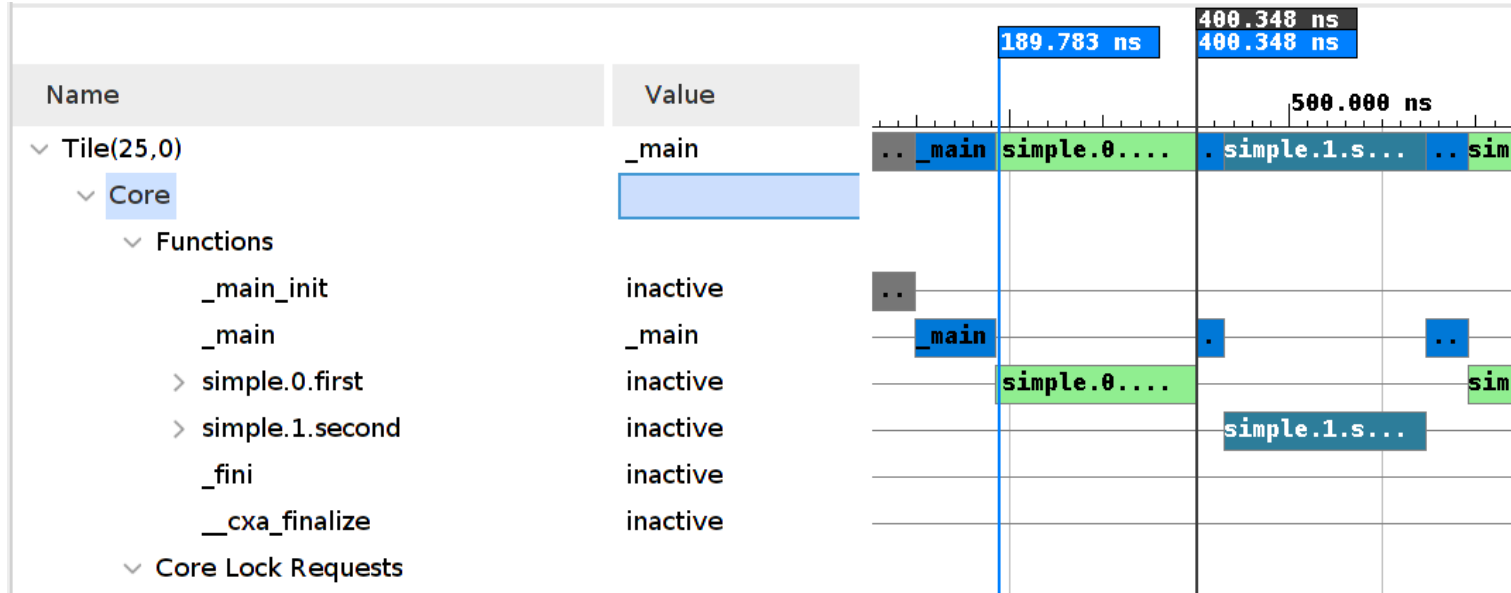
Vitis Analyzer - Trace



Starting out with the AI Engine tools



Vitis Analyzer



First kernel run frequency: $\frac{10^3}{(400.348 - 189.783) \times 10^{-9}} = 4.75 \text{ MHz}$

Run Cycles = $\frac{1250}{4.75} = 263 \text{ cycles}$

Starting out with the AI Engine tools



Vitis Analyzer

```
runtime<ratio>(first) = 0.1;  
runtime<ratio>(second) = 0.1;
```

Both kernels are estimated to run at 10% of the processing time of a single AI Engine

Run-time ratio = (cycles for one run of the kernel)/(cycle budget)

Cycle Budget = Block Size * (AI Engine Clock Frequency/Sample Frequency)

For example, take a kernel which processes a window of 128 samples and the input samples frequency (for example from an ADC) is 245.76MHz.

Cycle Budget is $128 * (1000 / 245.76) = 520$ cycles

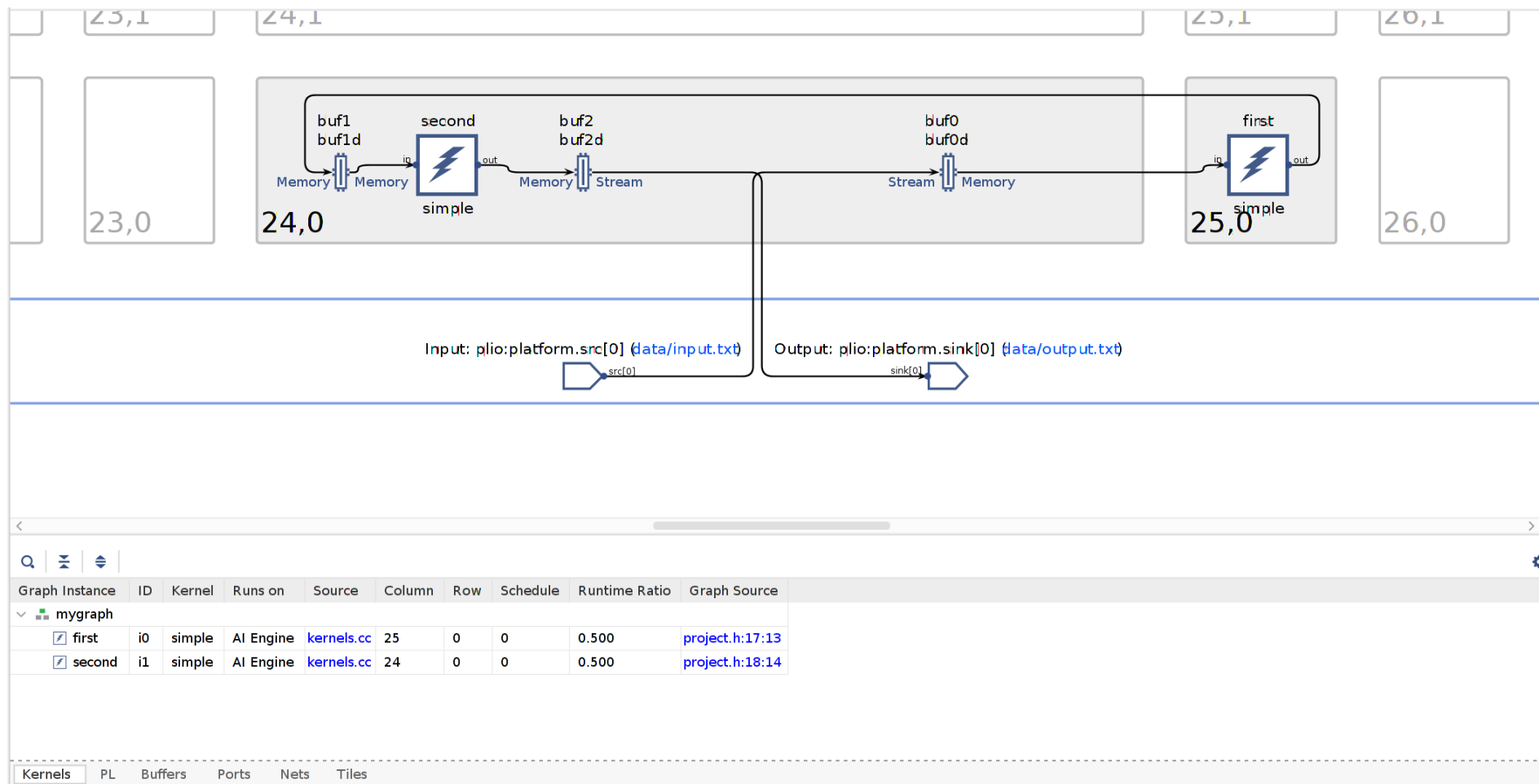
*** as above we can check at Vitis Analysis Chapter ***

Starting out with the AI Engine tools



Vitis Analyzer

```
runtime<ratio>(first) = 0.1;  Modify ratio number to 0.5  
runtime<ratio>(second) = 0.1;
```



Starting out with the AI Engine tools



Output File Timing Stamp

```
runtime<ratio>(first) = 0.1;  Modify ratio number to 0.5  
runtime<ratio>(second) = 0.1;
```

```
project.h  project.cpp  kernels.cc  output.txt  2  
54 10 8  
55 T 774400 ps  
56 14 12  
57 T 777600 ps  
58 18 16  
59 T 780800 ps  
60 22 20  
61 T 784 ns  
62 26 24  
63 T 787200 ps  
64 TLAST  
65 30 28  
66 T 963200 ps  
67 0 2  
68 T 966400 ps  
69 4 6  
70 T 969600 ps  
71 8 10  
72 T 972800 ps  
73 12 14  
74 T 976 ns  
75 16 18  
76 T 979200 ps  
77 20 22  
78 T 982400 ps  
79 24 26  
80 T 985600 ps  
81 28 30  
82 T 988800 ps  
83 2 0  
84 T 992 ns  
85 6 4  
output2.txt 2  
54 10 8  
55 T 761600 ps  
56 14 12  
57 T 764800 ps  
58 18 16  
59 T 768 ns  
60 22 20  
61 T 771200 ps  
62 26 24  
63 T 774400 ps  
64 TLAST  
65 30 28  
66 T 1184 ns  
67 0 2  
68 T 1187200 ps  
69 4 6  
70 T 1190400 ps  
71 8 10  
72 T 1193600 ps  
73 12 14  
74 T 1196800 ps  
75 16 18  
76 T 1200 ns  
77 20 22  
78 T 1203200 ps  
79 24 26  
80 T 1206400 ps  
81 28 30  
82 T 1209600 ps  
83 2 0  
84 T 1212800 ps  
85 6 4
```


Starting out with the AI Engine tools



Output File Timing Stamp

```
runtime<ratio>(first) = 0.1; Modify ratio number to 0.5  
runtime<ratio>(second) = 0.1;
```

```
project.h project.cpp kernels.cc output.txt x2  
119 10 8  
120 T 1049600 ps  
121 14 12  
122 T 1052800 ps  
123 18 16  
124 T 1056 ns  
125 22 20  
126 T 1059200 ps  
127 26 24  
128 T 1062400 ps  
129 TLAST  
130 30 28  
131 T 1241600 ps  
132 0 2  
133 T 1244800 ps  
134 4 6  
135 T 1248 ns  
136 8 10  
137 T 1251200 ps  
138 12 14  
139 T 1254400 ps  
140 16 18  
141 T 1257600 ps  
142 20 22  
143 T 1260800 ps  
144 24 26  
145 T 1264 ns  
146 28 30  
147 T 1267200 ps  
148 2 0  
149 T 1270400 ps  
150 6 4  
output2.txt x2  
119 10 8  
120 T 1270400 ps  
121 14 12  
122 T 1273600 ps  
123 18 16  
124 T 1276800 ps  
125 22 20  
126 T 1280 ns  
127 26 24  
128 T 1283200 ps  
129 TLAST  
130 30 28  
131 T 1692800 ps  
132 0 2  
133 T 1696 ns  
134 4 6  
135 T 1699200 ps  
136 8 10  
137 T 1702400 ps  
138 12 14  
139 T 1705600 ps  
140 16 18  
141 T 1708800 ps  
142 20 22  
143 T 1712 ns  
144 24 26  
145 T 1715200 ps  
146 28 30  
147 T 1718400 ps  
148 2 0  
149 T 1721600 ps  
150 6 4
```

Starting out with the AI Engine tools



Data Access Mechanisms

The kernels from the template we are looking at are accessing data using windows. Another type of access available for AIE kernels is stream access.

Window Based Access

When a kernel uses a window based access, it reads directly from the local memory of the AI Engine it is running on or the local memory of one of the neighboring AI Engines.

The kernel is loaded in the AI Engine only when the input window is full. This means that the first invocation of the kernel will have an initial latency while the input window is filled in. However, as the memory can be used as a ping pong buffer, the next set of data can be written into memory during the kernel execution and be ready for the next iteration.

Stream Based Access

When a kernel uses a stream based access, it reads directly from the AXI-Stream interface. In this case the kernel is reading the data in a sample by sample fashion.

Using streams can introduce back pressure to the upstream kernels if the downstream kernel is not able to process the data fast enough but can also create a stall in a downstream kernel if the upstream kernel is not able to produce data fast enough.

Vitis AIE API

Wait for Updating.....

Most memory access functions in the AIE API accept an enum value from `aie_dm_resource` that can be used to bind individual accesses to a virtual resource.

```
void fn(int __aie_dm_resource_a * A,  
int * B, *D,  
int __aie_dm_resource_a * C)  
{  
    aie::vector<int,8> v1 = aie::load_v<8>(A); // Access from A and C are bound to the same virtual resource so they  
    aie::vector<int,8> v2 = aie::load_v<8>(B); // are never scheduled on the same instruction. B is not annotated so  
    aie::vector<int,8> v3 = aie::load_v<8>(C); // its memory accesses can be scheduled in the same instruction with  
                                              // accesses to A or C.  
    aie::vector<int,8> v3 = aie::load_v<8, aie_dm_resource::b>(D); // Memory access can be annotated  
                                                                    // in load operation  
  
    ...  
}
```

Disclaimer and Attribution

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's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

© Copyright 2022 Advanced Micro Devices, Inc. All rights reserved. Xilinx, the Xilinx logo, AMD, the AMD Arrow logo, Alveo, Artix, Kintex, Kria, Spartan, Versal, Vitis, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.



APPENDIX: ISSUE

asm/errno.h: No such file or directory

Solution:

In -s /usr/include/asm-generic /usr/include/asm

ERROR: [aiecompiler 77-753]

Solution:

Get AIE License from Xilinx website

[Licensing Solution Center \(xilinx.com\)](https://www.xilinx.com/au/aielicensing.html)