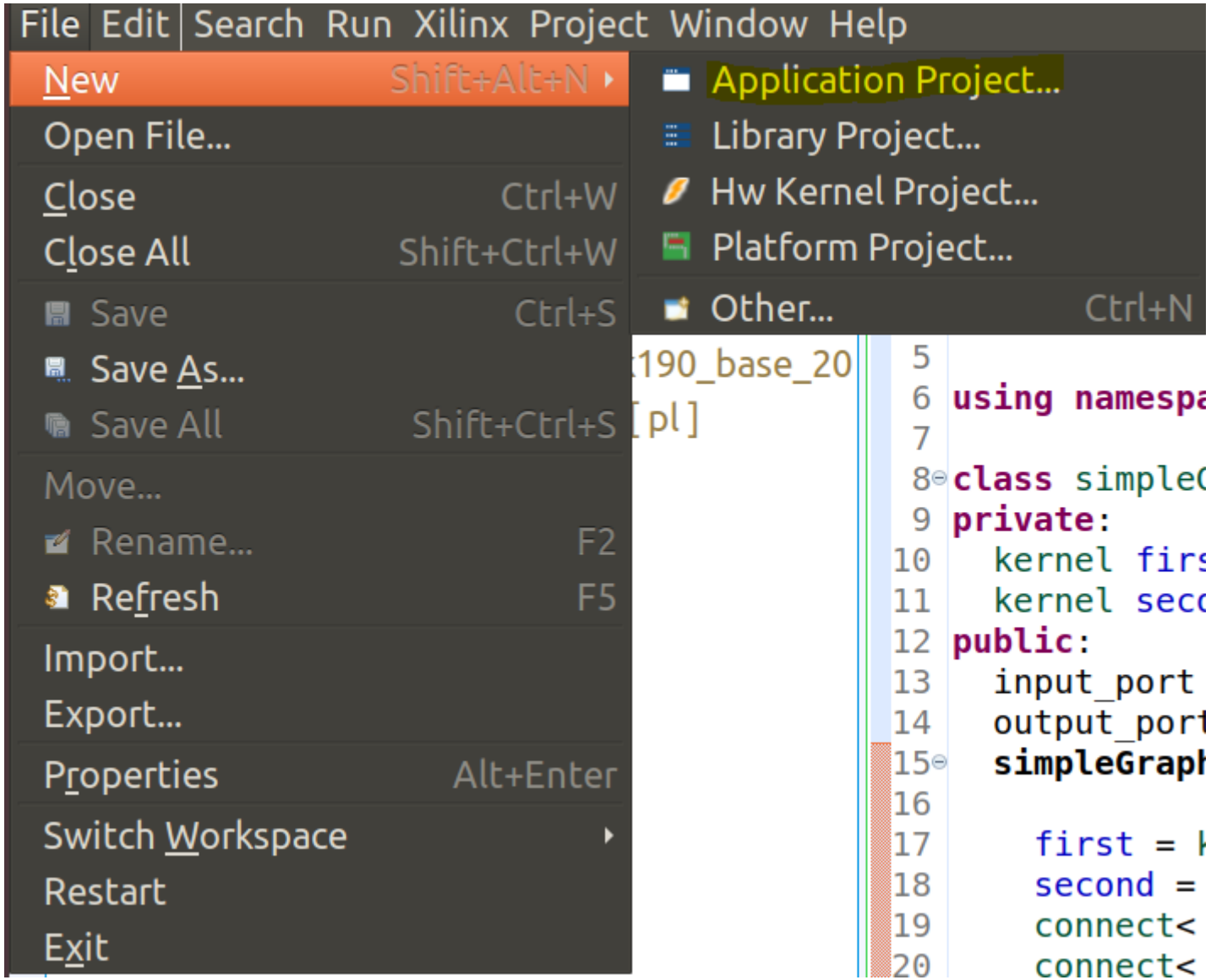
AMD
XILINX

Versal ACAP: Vitis Tool Flow

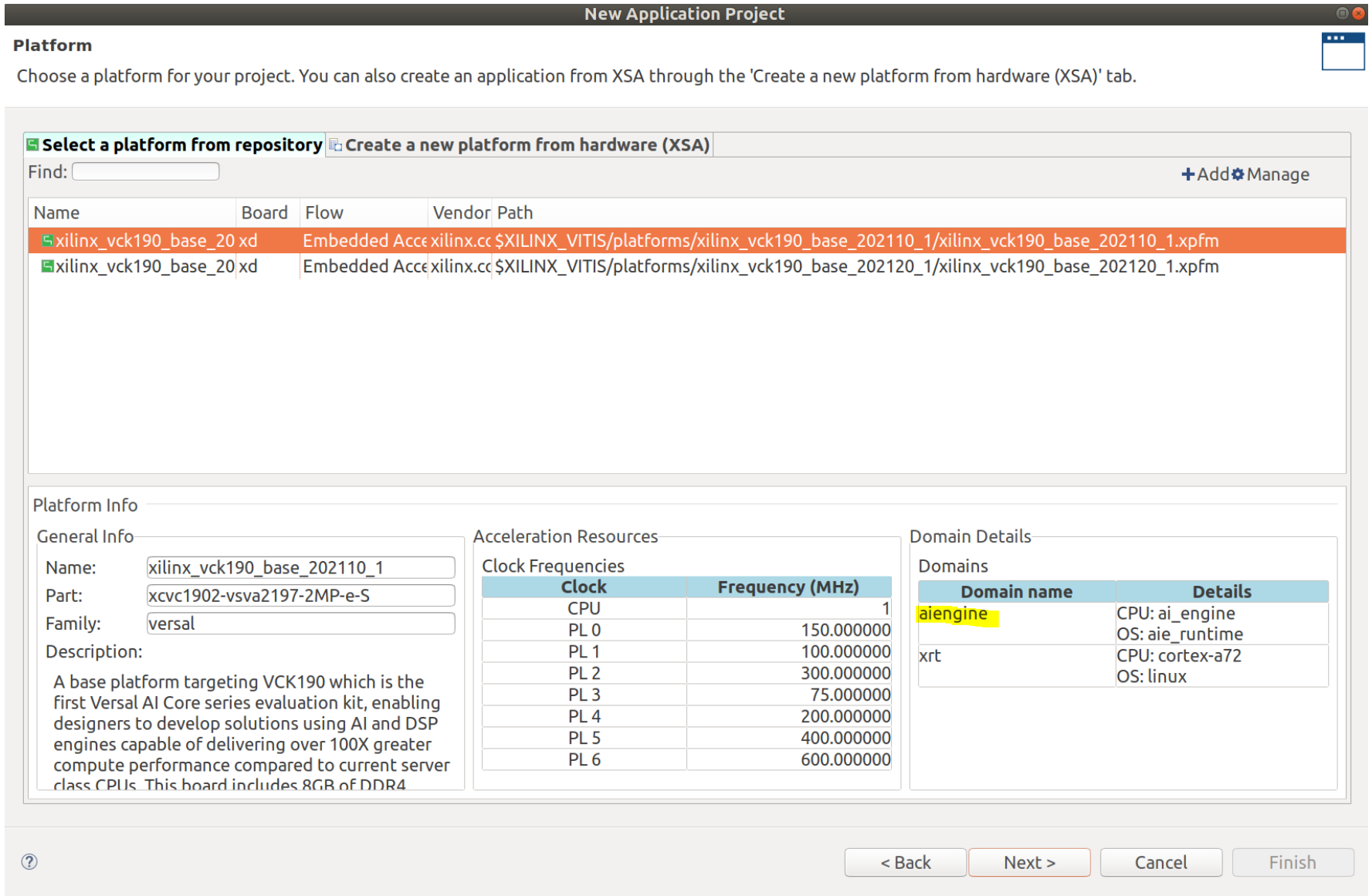
Revision History

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

Starting out with the AI Engine tools



Starting out with the AI Engine tools



Starting out with the AI Engine tools

XILINX VITIS

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)

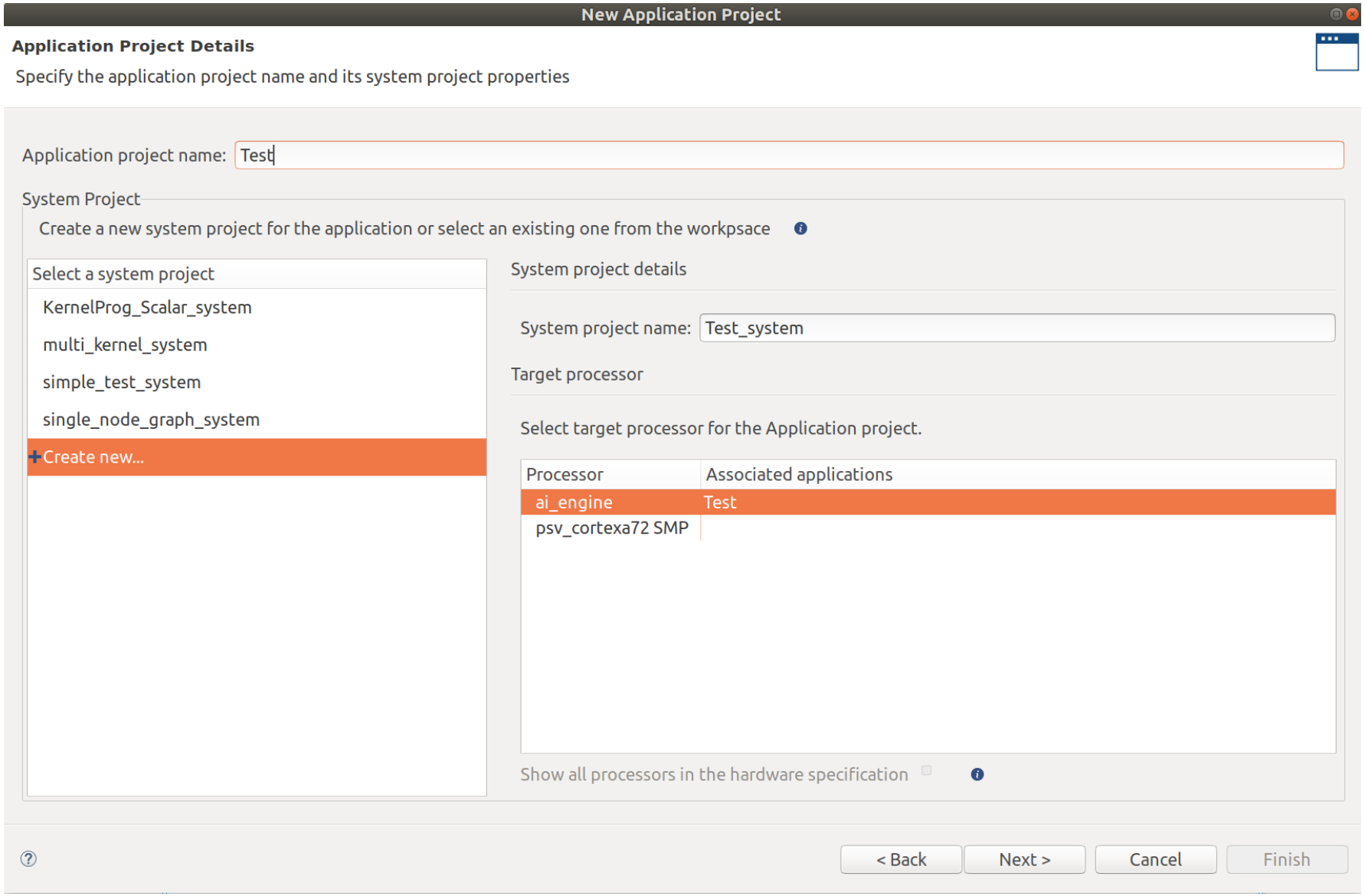
MD5 SUM Value : a57b232c230669a54b718103c2cde12a

VMK180 Base 2021.1 (ZIP - 23.41 MB)

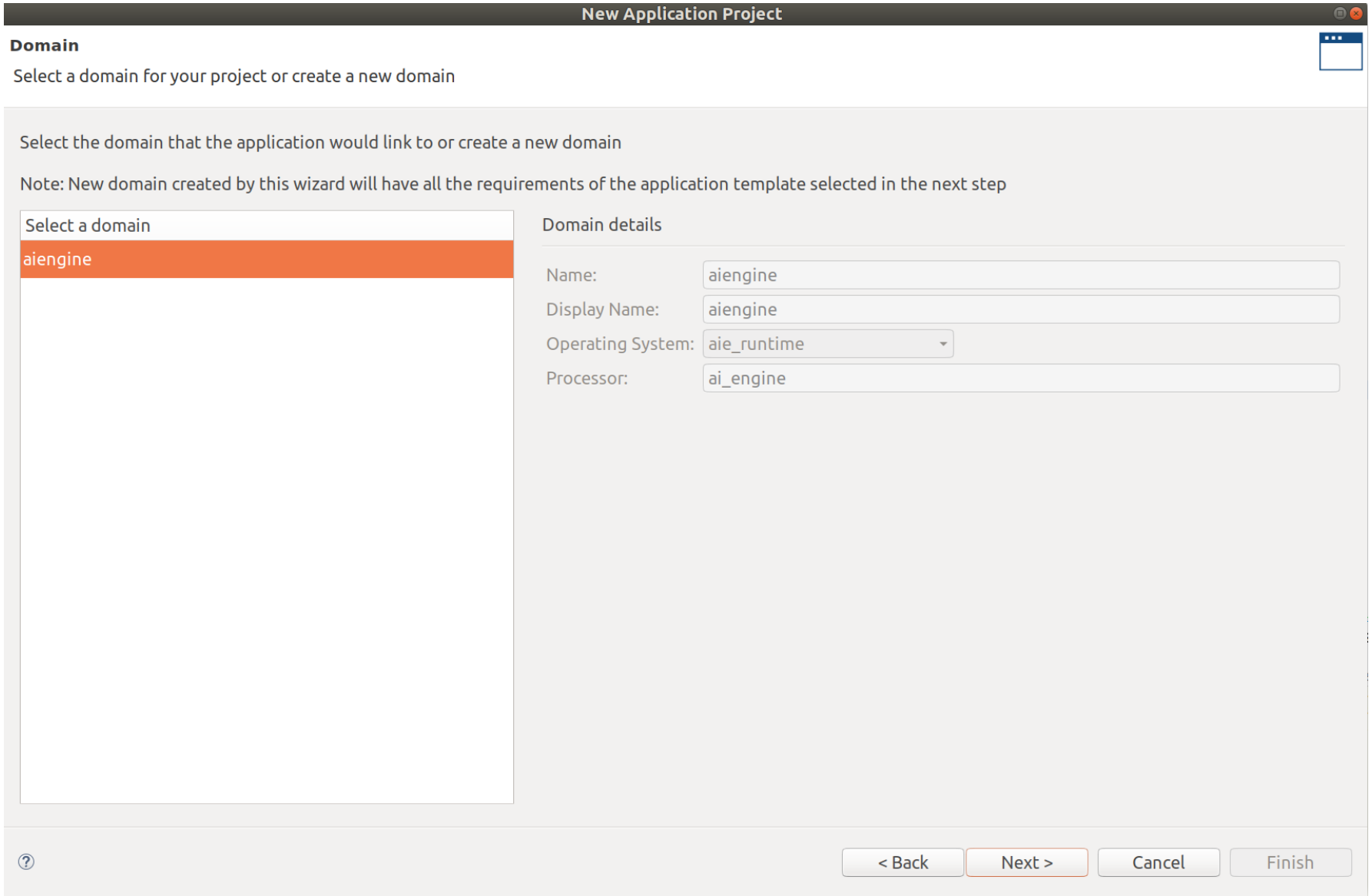
MD5 SUM Value : d4f3d7c7cf01da399831bdacd67c3733

AMD XILINX

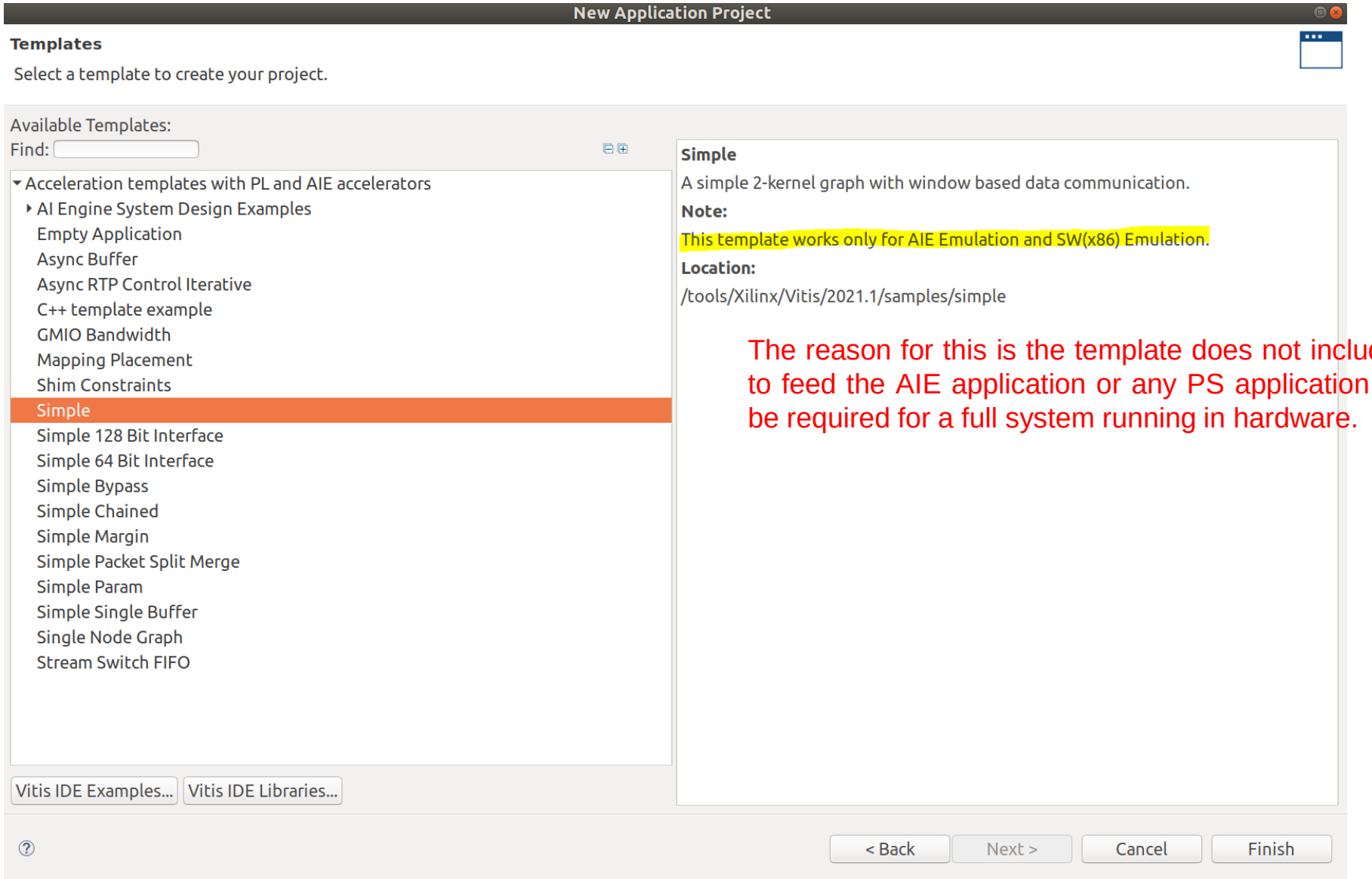
Starting out with the AI Engine tools



Starting out with the AI Engine tools

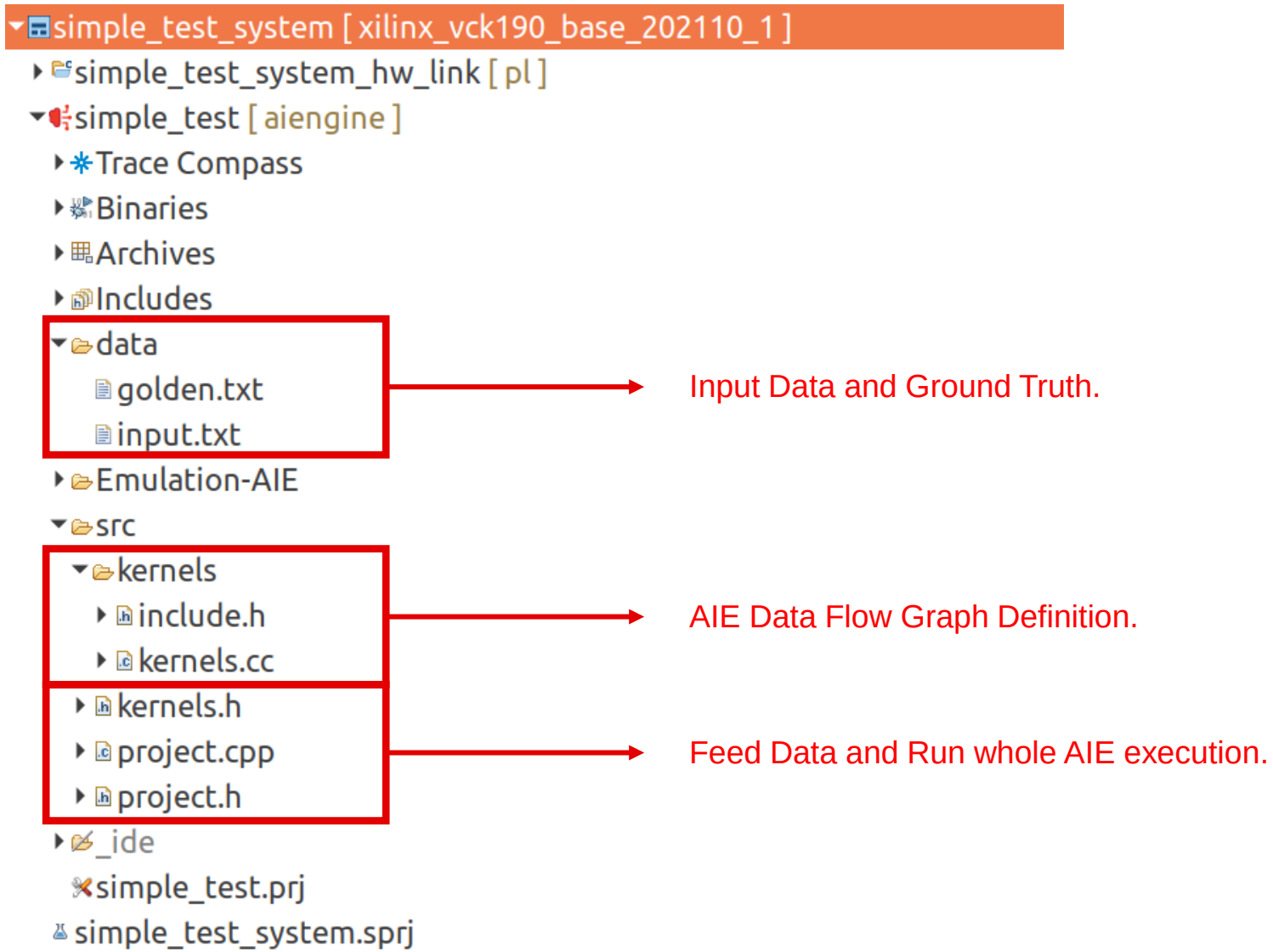


Starting out with the AI Engine tools



The reason for this is the template does not include any I/O to feed the AIE application or any PS application which will be required for a full system running in hardware.

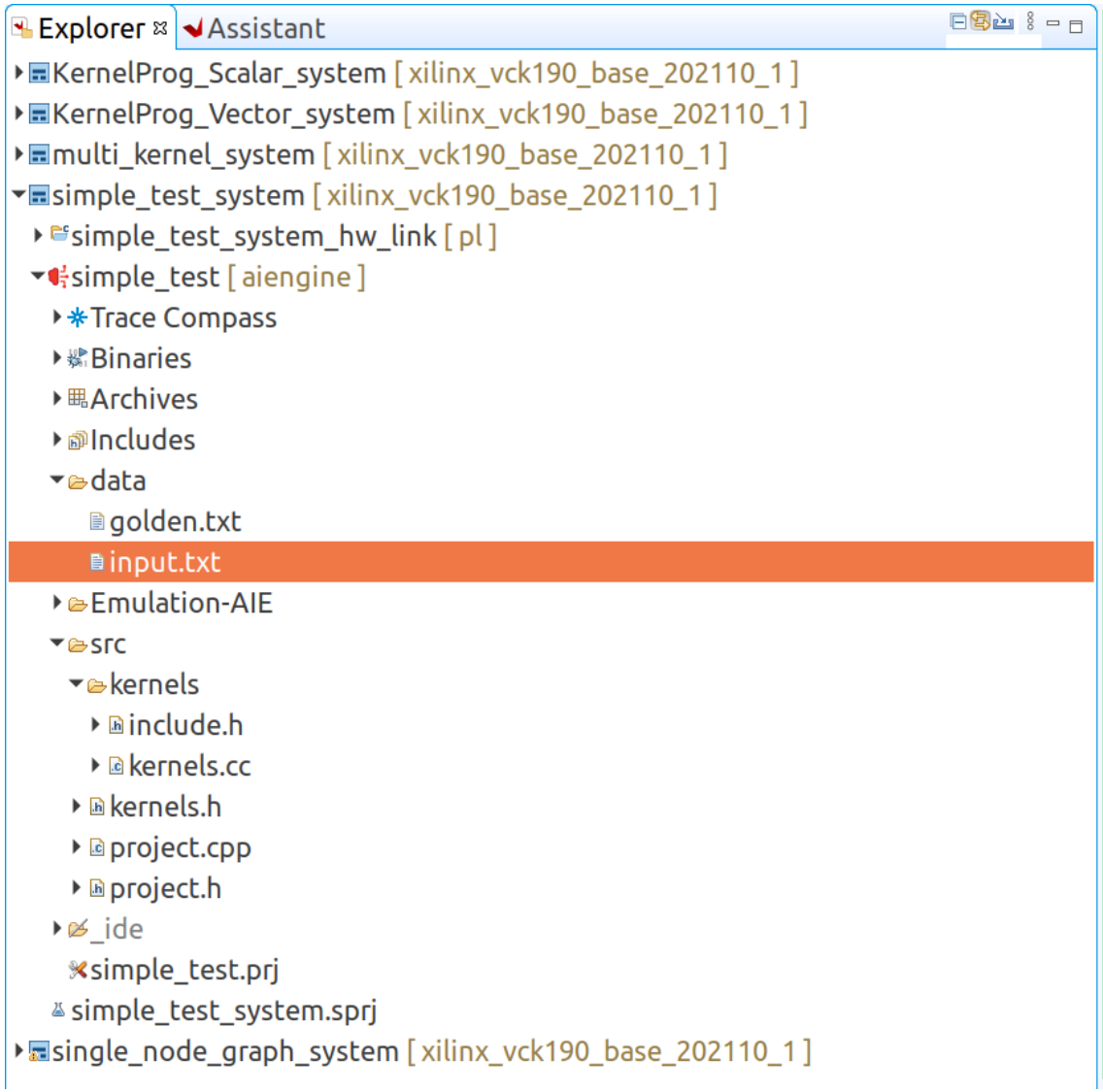
Starting out with the AI Engine tools

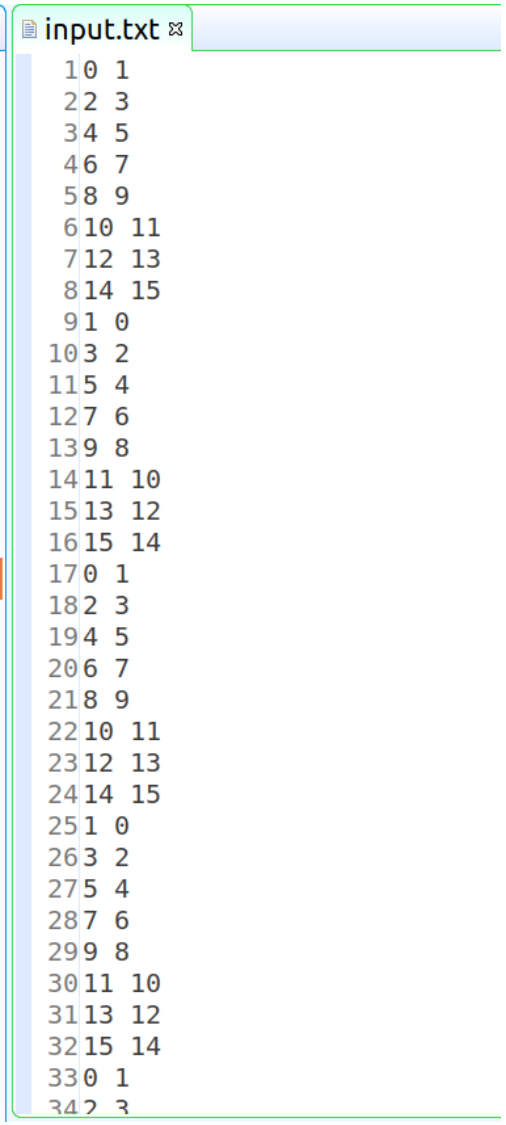


Starting out with the AI Engine tools

XILINX VITIS

Input Data



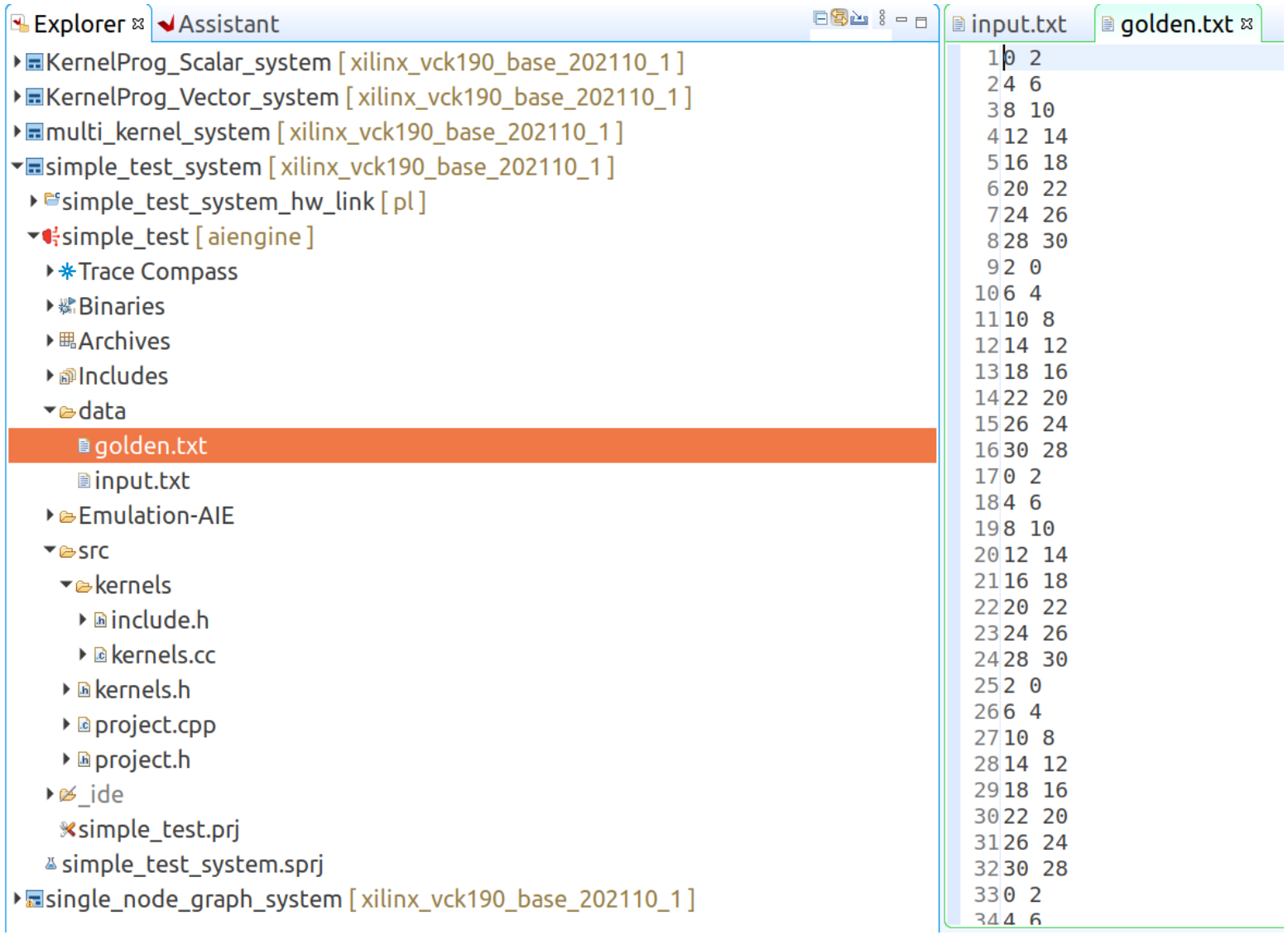


AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Ground Truth

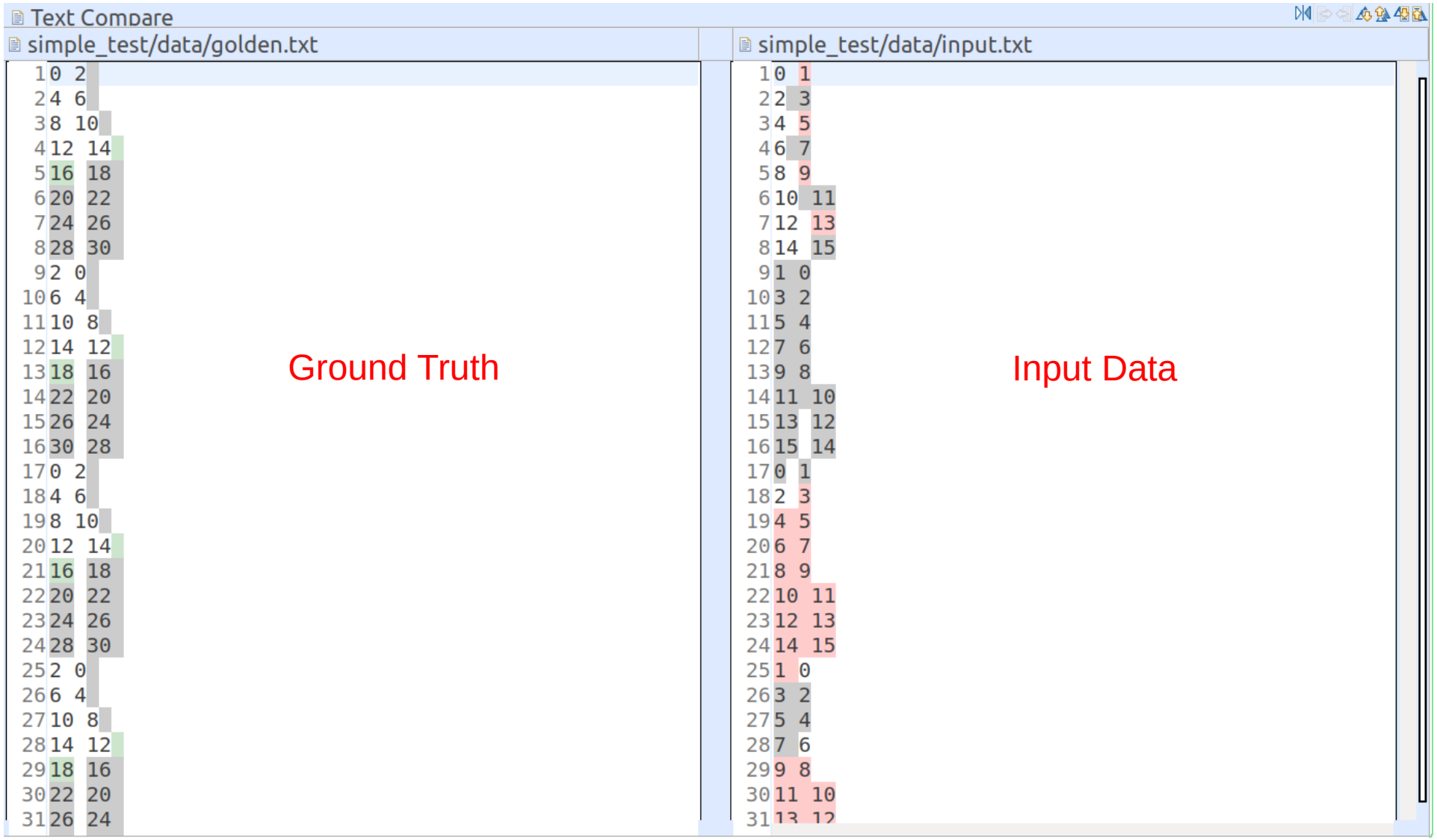


AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Text Comparation

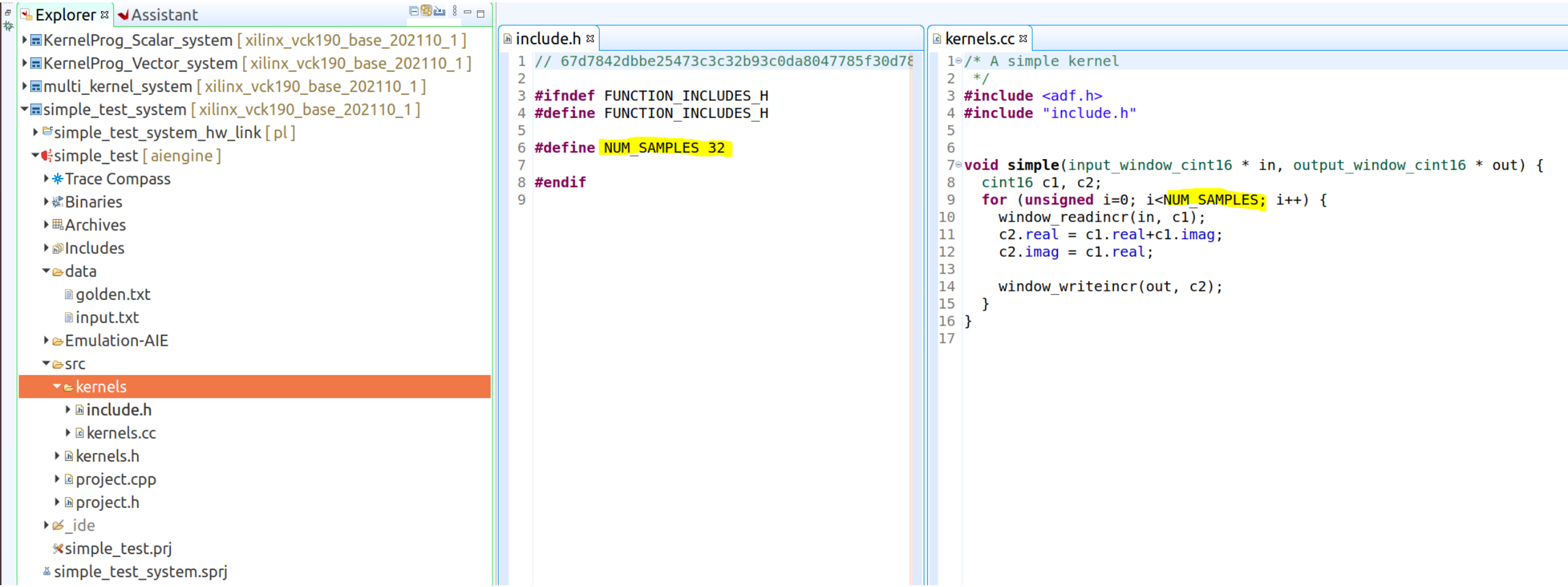


AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

AIE Data Flow Graph Definition



AMD XILINX

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

XILINX VITIS

AIE Data Flow Graph Definition

Table 34: **Data Type and PLIO Width**

Data Type	PLIO 32 bit `adf::input_plio in = adf::input_plio::create("DataIn1", adf::plio_32_bits, "input.csv");`	PLIO 64 bit `adf::input_plio in = adf::input_plio::create("DataIn1", adf::plio_64_bits, "input.csv");`	PLIO 128 bit `adf::input_plio in = adf::input_plio::create("DataIn1", adf::plio_128_bits, "input.csv");`

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, 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,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 `cint` value (real, imag). For example:
	`CMD, D,D,TKEEP, TLAST`
	`DATA,1980,485,-1,0`
		Two `cint` 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 `cint` 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 `cint` value (real, imag) represented by two data columns expected. For example:
		`CMD, D, D, TKEEP, TLAST`
		`DATA, 1980, 485, -1, 0`
			Two `cint` 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
```

AMD XILINX

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

XILINX VITIS

Feed Data and Run whole AIE execution

AMD XILINX

Starting out with the AI Engine tools

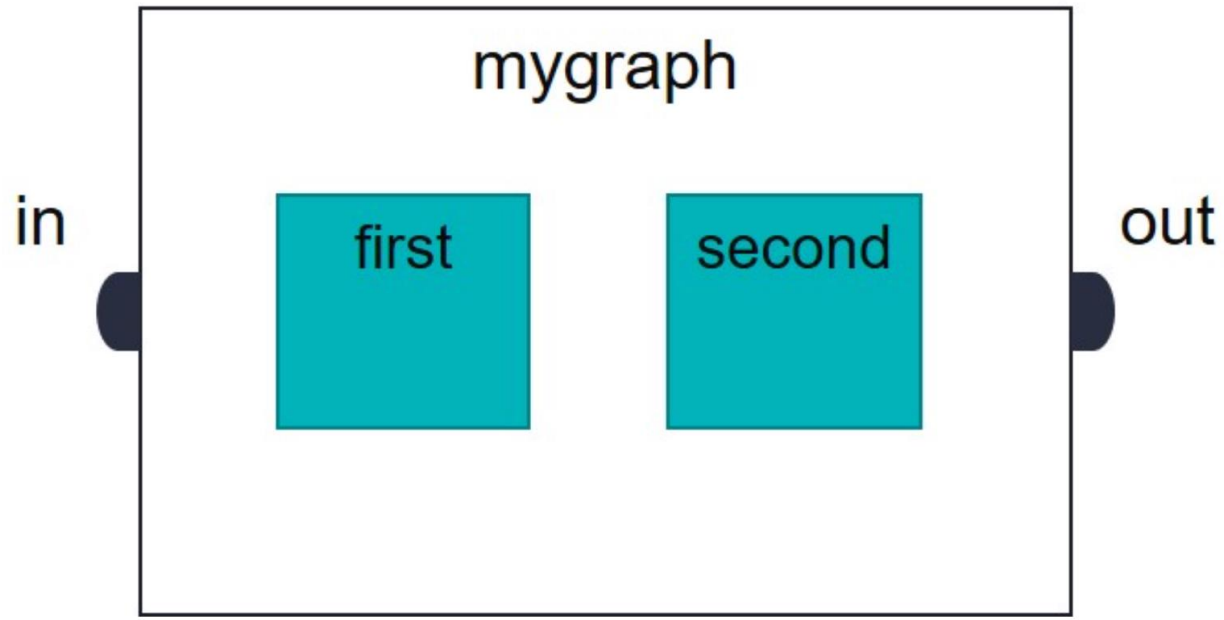
XILINX VITIS

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



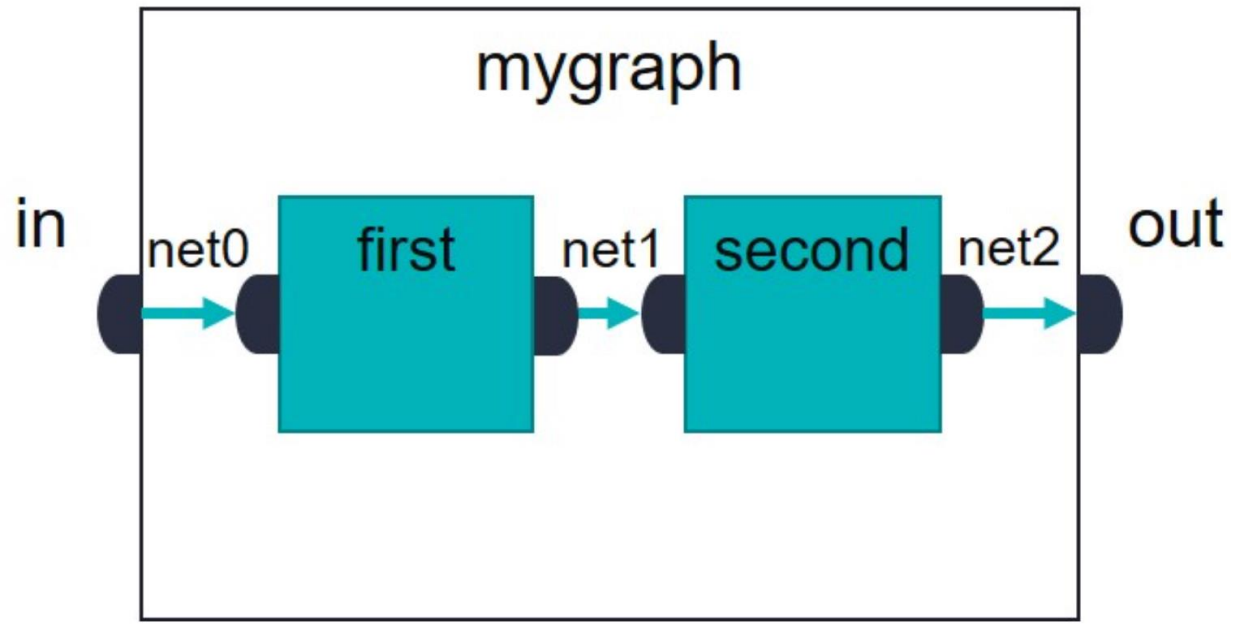
AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

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);
```



AMD XILINX

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

AMD
XILINX

Starting out with the AI Engine tools

XILINX VITIS

Feed Data and Run whole AIE execution

project.h | project.cpp

```
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
```

Define Real Data input and output file

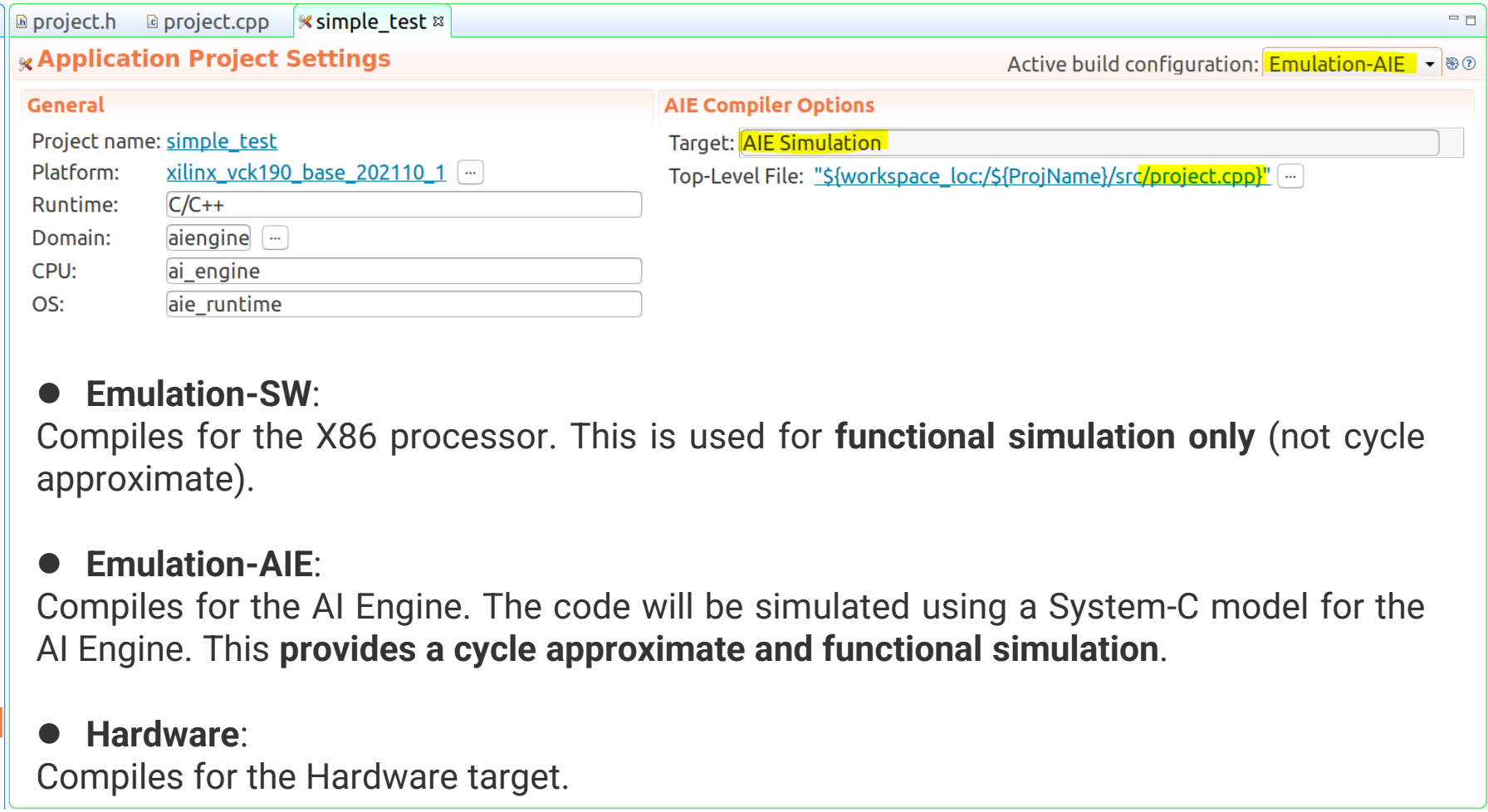
Run for 4 iteration

AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Build and Run Project



- **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**.

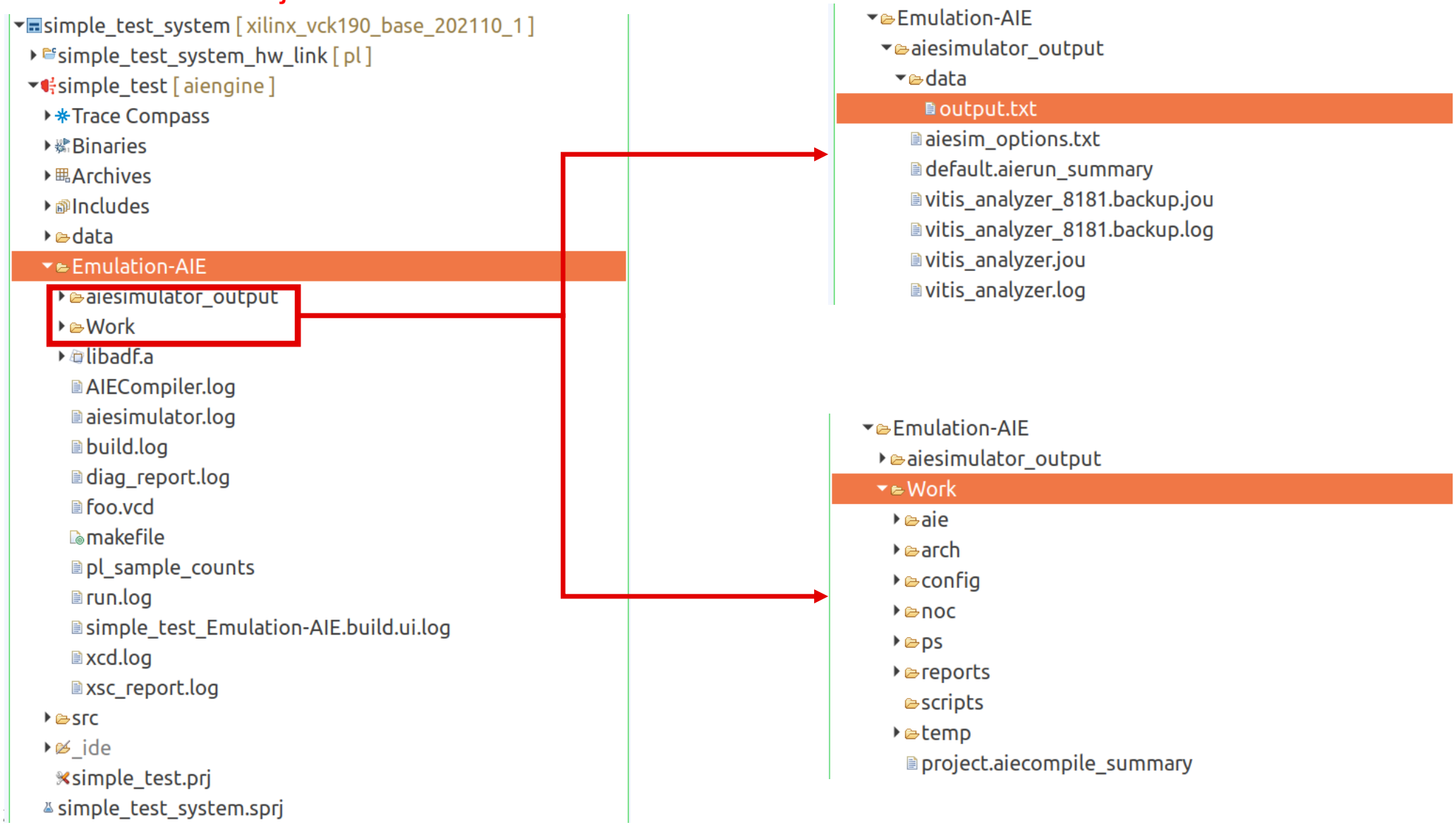
- **Hardware**:
Compiles for the Hardware target.

AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Build and Run Project

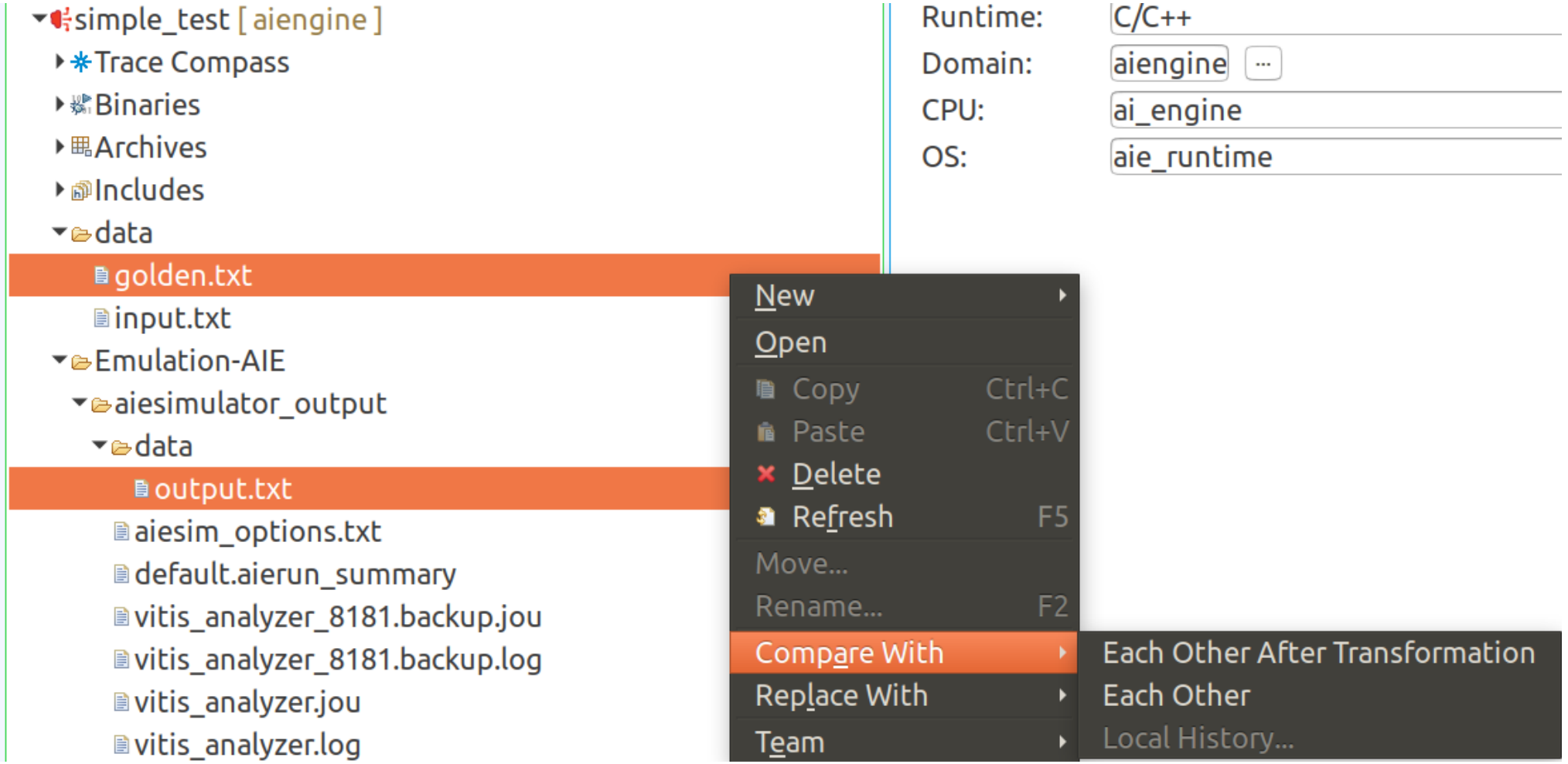


AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Verify Output and check AIE execution status

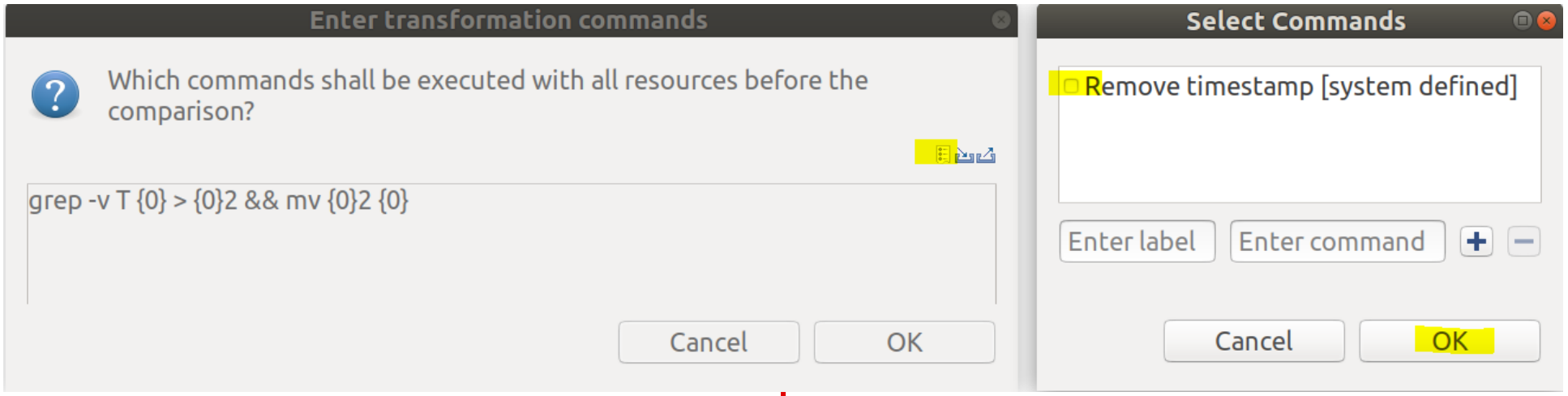


AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Verify Output and check AIE execution status



simple_test/data/golden.txt

```
0 2
4 6
8 10
12 14
16 18
20 22
24 26
28 30
2 0
6 4
10 8
14 12
18 16
22 20
26 24
```

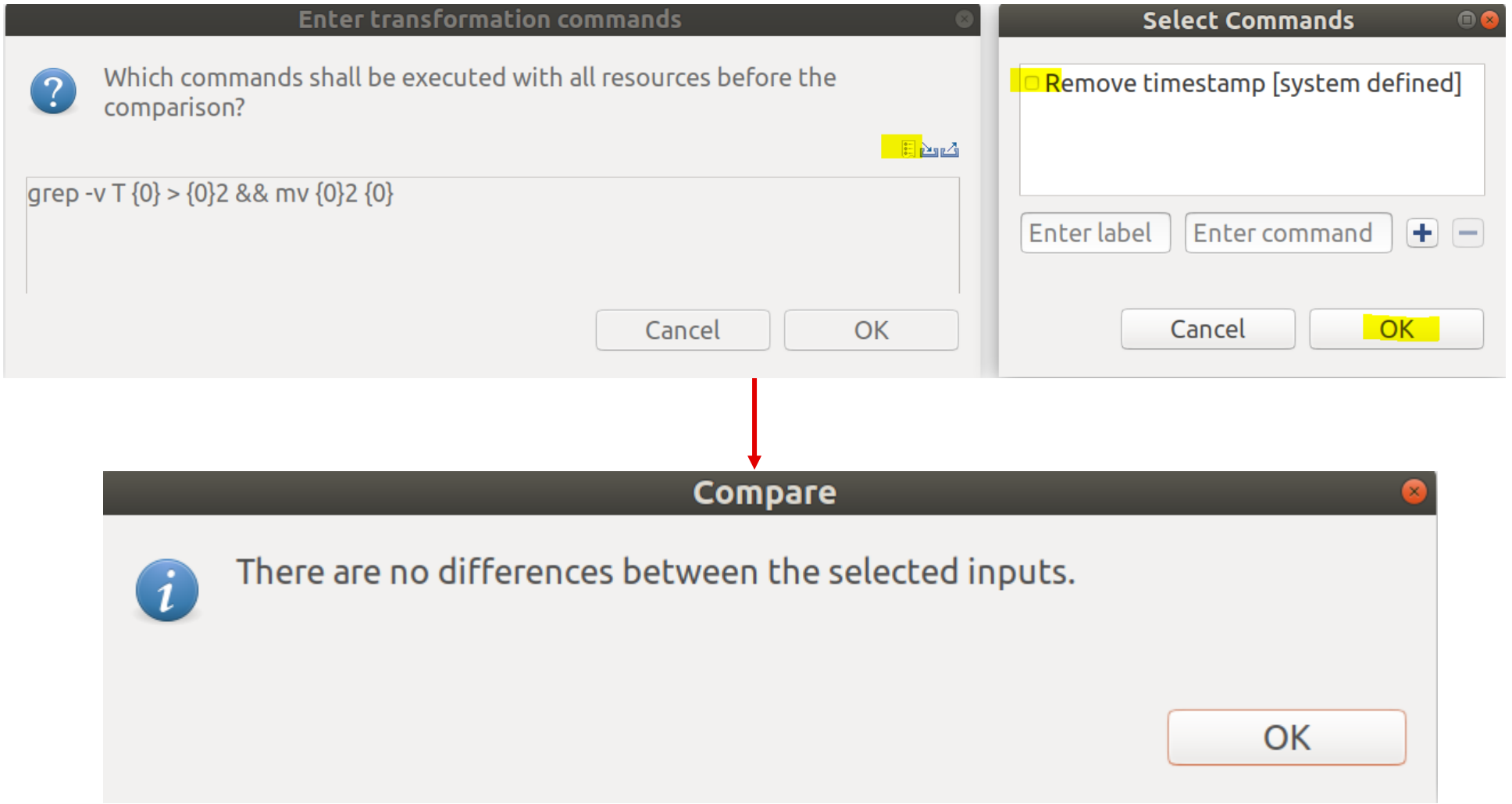
simple_test/Emulation-AIE/aiesimulator_output/data/output.txt

```
1 1
7 5
13 9
19 13
25 17
31 21
37 25
43 29
2 1
8 5
14 9
20 13
26 17
32 21
38 25
```

AMD XILINX

Starting out with the AI Engine tools

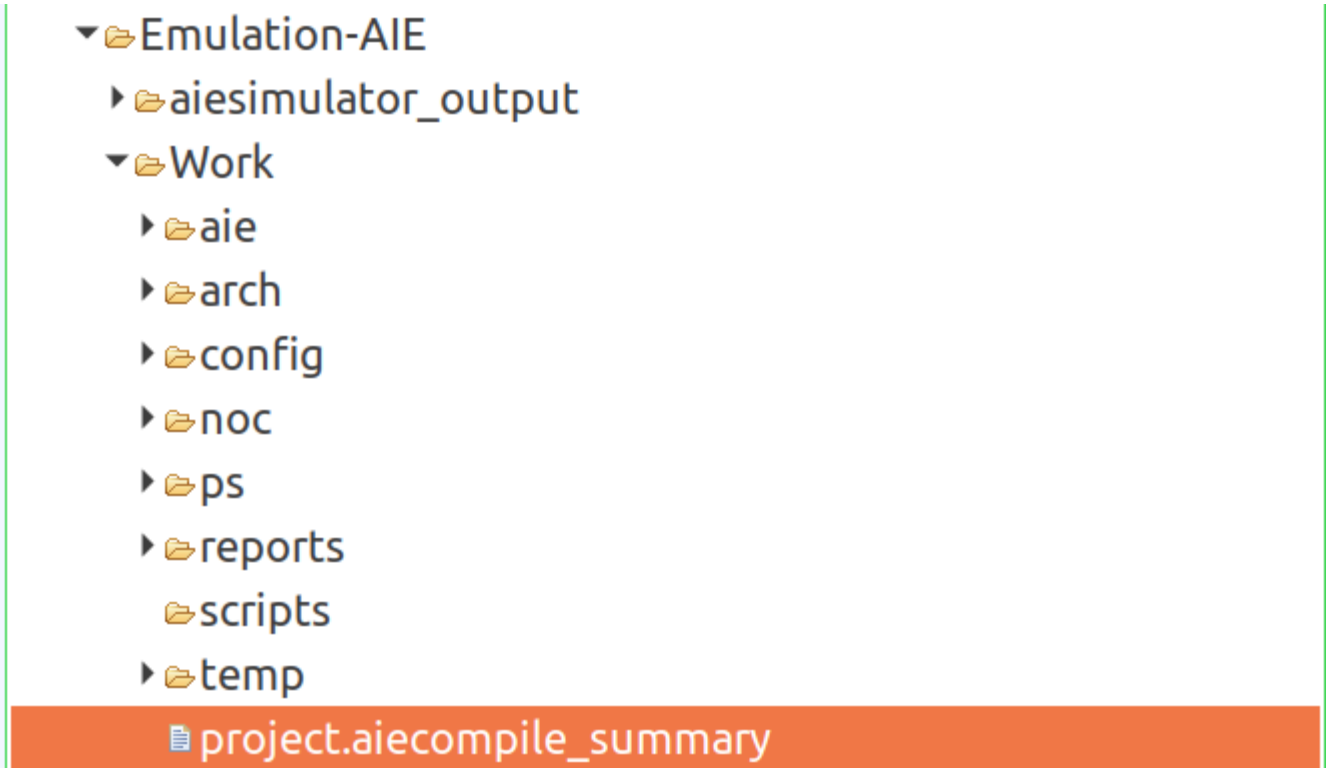
Verify Output and check AIE execution status



Starting out with the AI Engine tools

XILINX VITIS

Verify Output and check AIE execution status

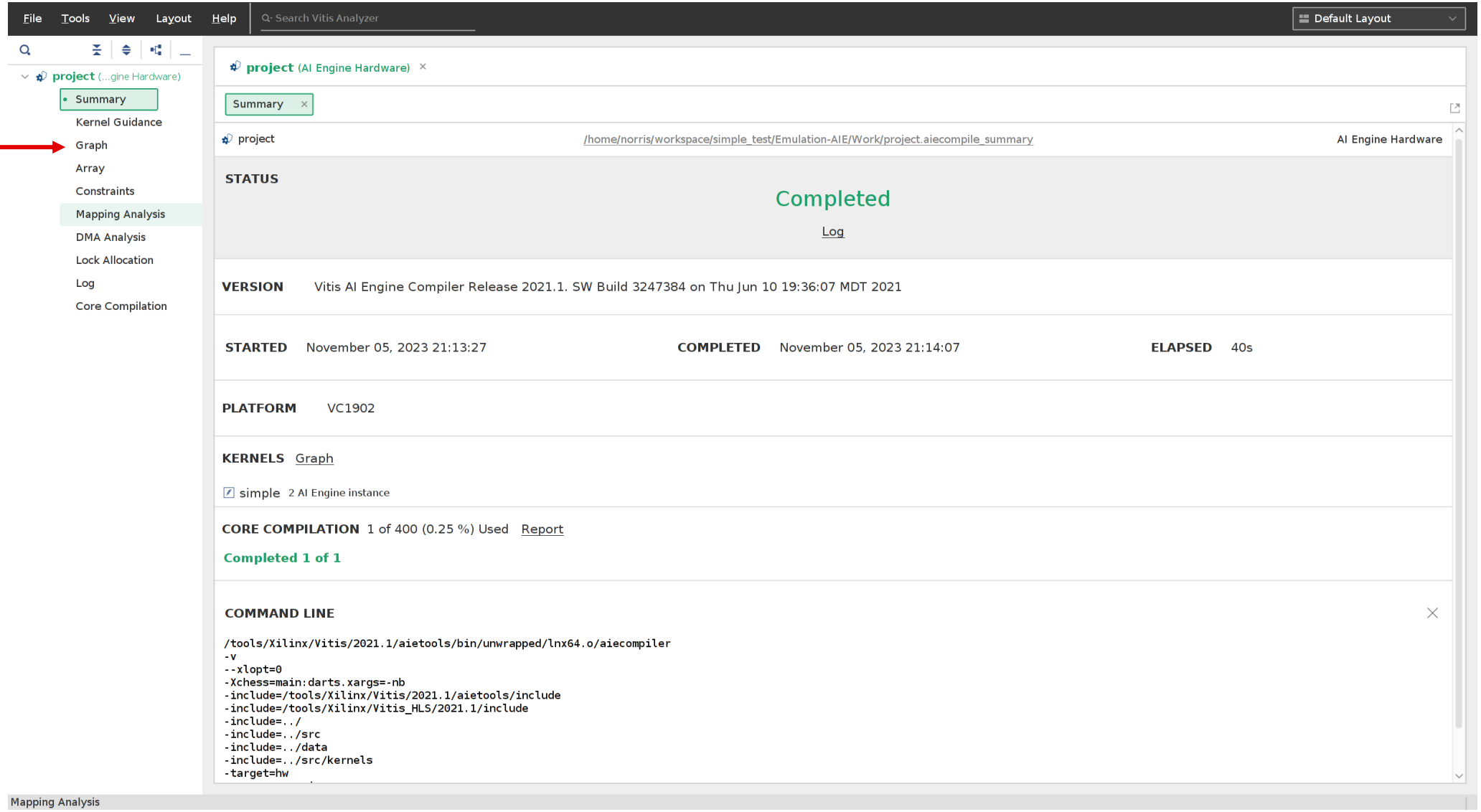


Double Click

AMD XILINX

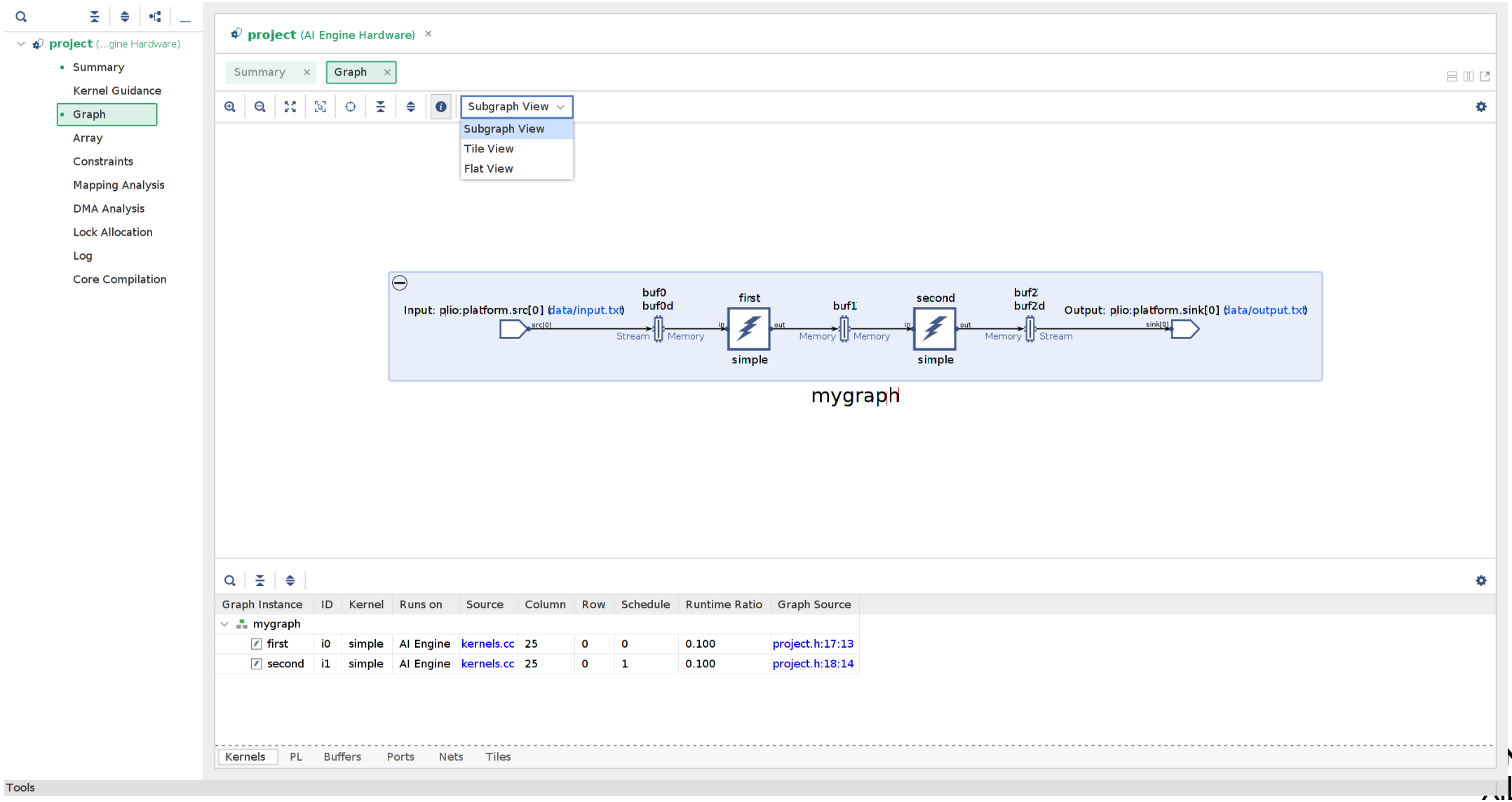
Starting out with the AI Engine tools

Vitis Analyzer



Starting out with the AI Engine tools

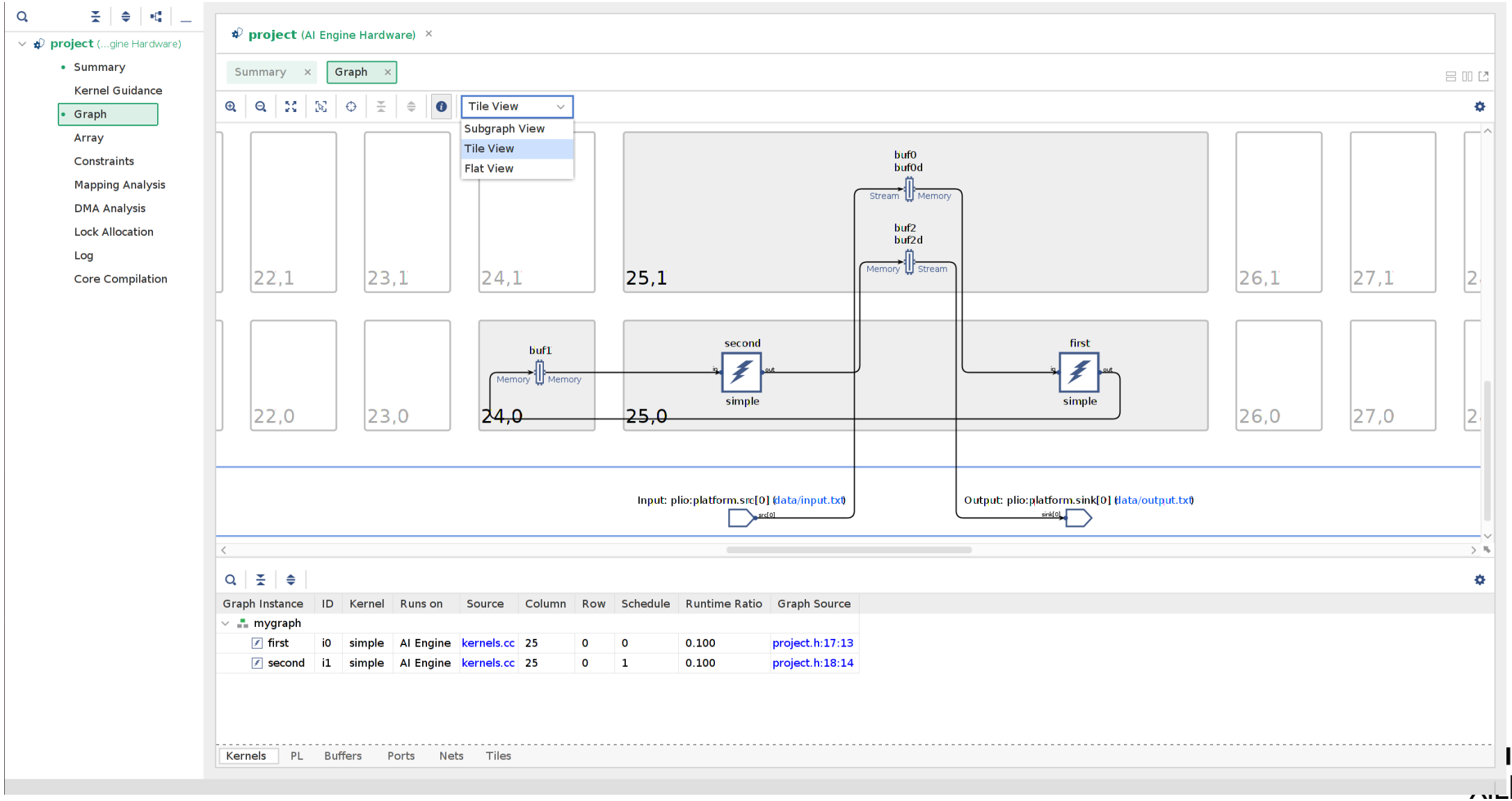
Vitis Analyzer



Starting out with the AI Engine tools

XILINX VITIS

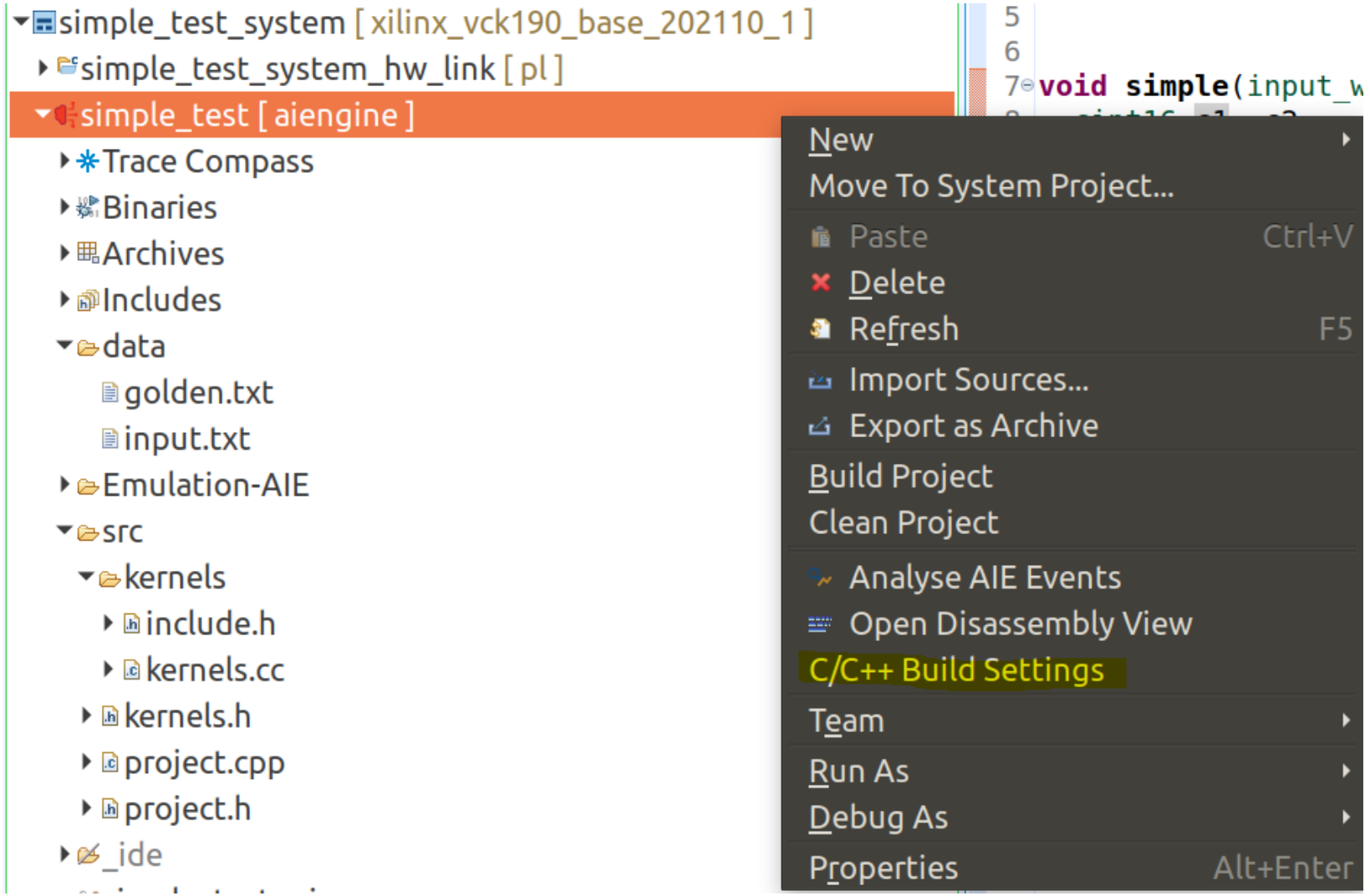
Vitis Analyzer



Starting out with the AI Engine tools

XILINX VITIS

Verify Output and check AIE execution status

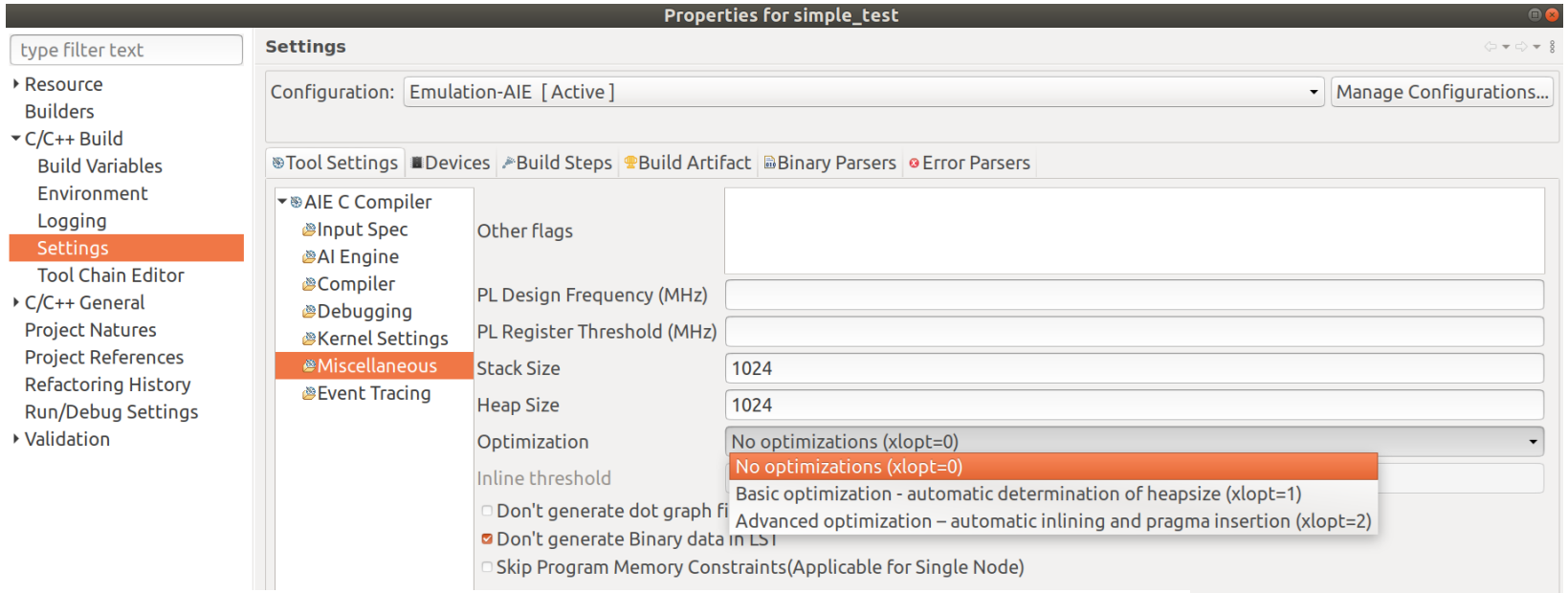


AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Verify Output and check AIE execution status



Optimization Options

- **xlopt:**

  Enable kernel optimizations based on the specified value:

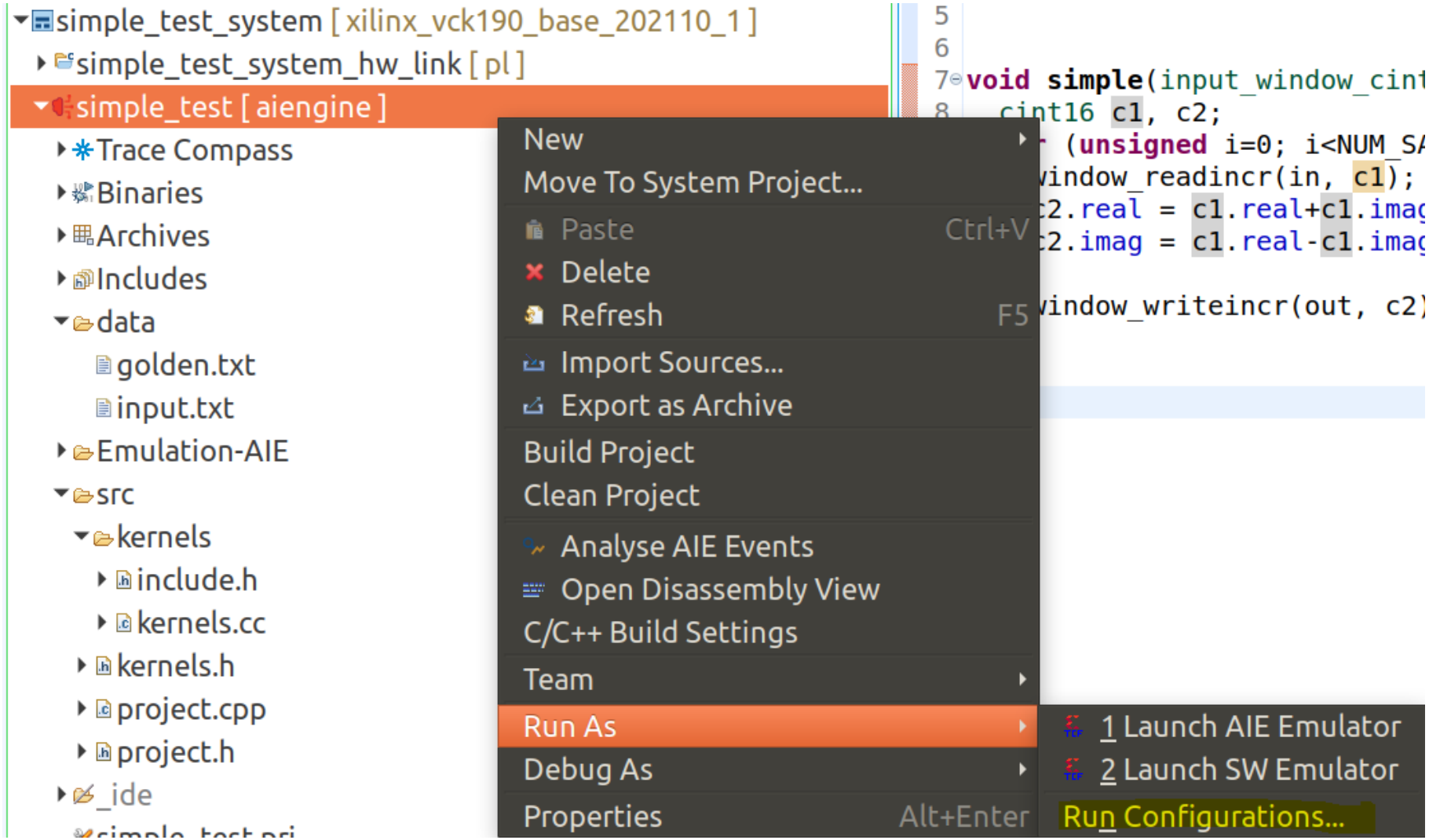
  - 0: No kernel optimizations.
  - 1: Computation of heap size, generation of guidance based on LLVM IR analysis, and insertion of loop pragmas.
  - 2: The same optimizations as 1 with the addition of loop peeling for unrolled loops, and automatic inlining.

AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Verify Output and check AIE execution status

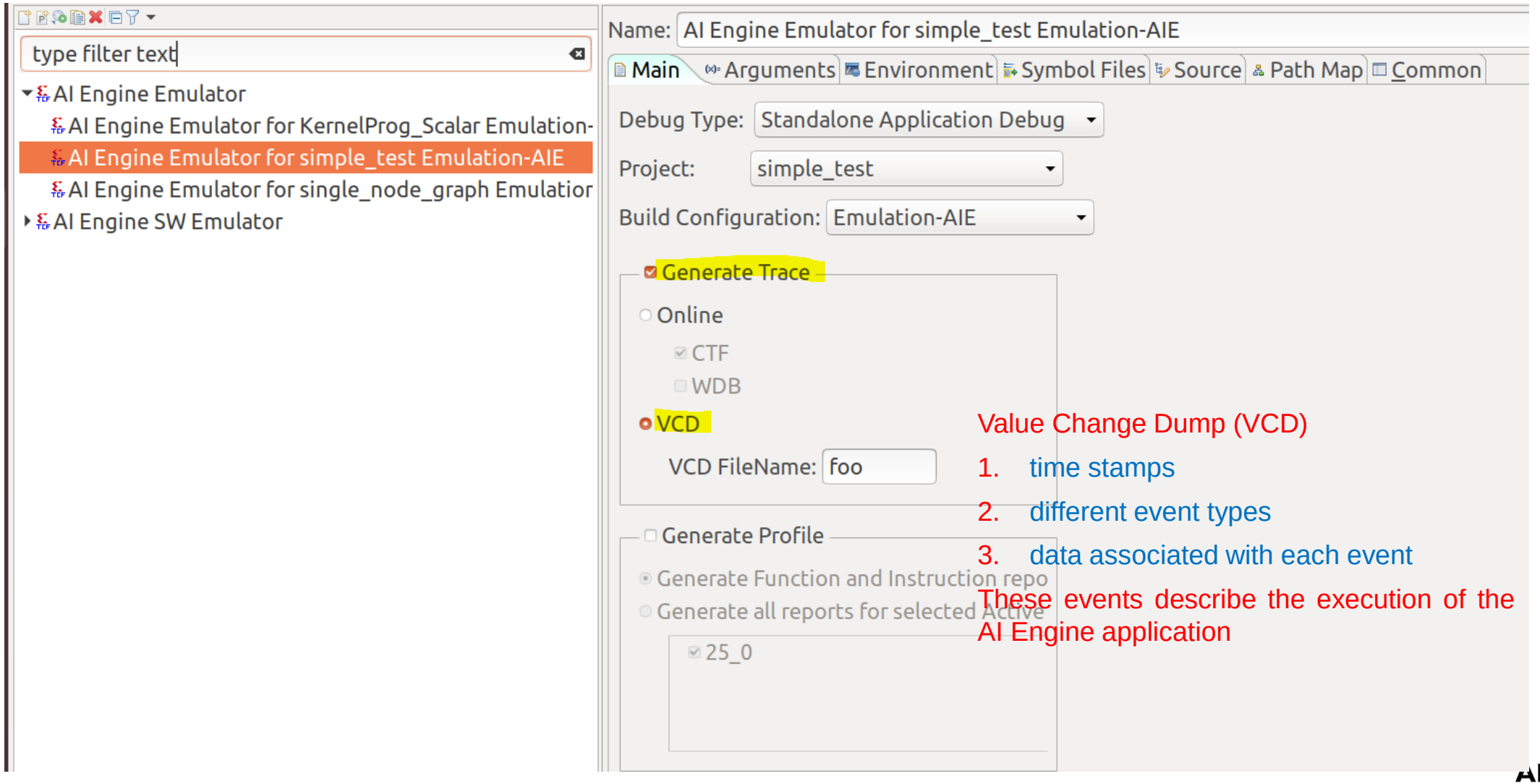


AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Verify Output and check AIE execution status – rerun the project



Value Change Dump (VCD)

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

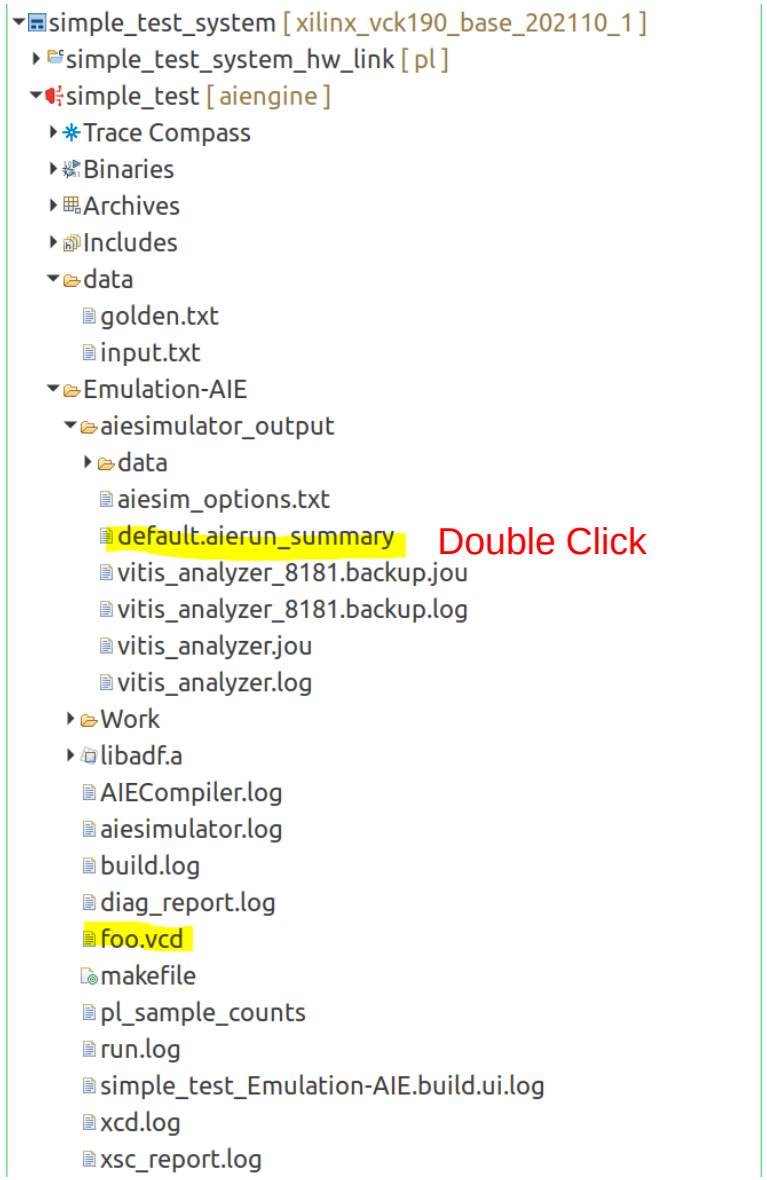
These events describe the execution of the AI Engine application

AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Verify Output and check AIE execution status

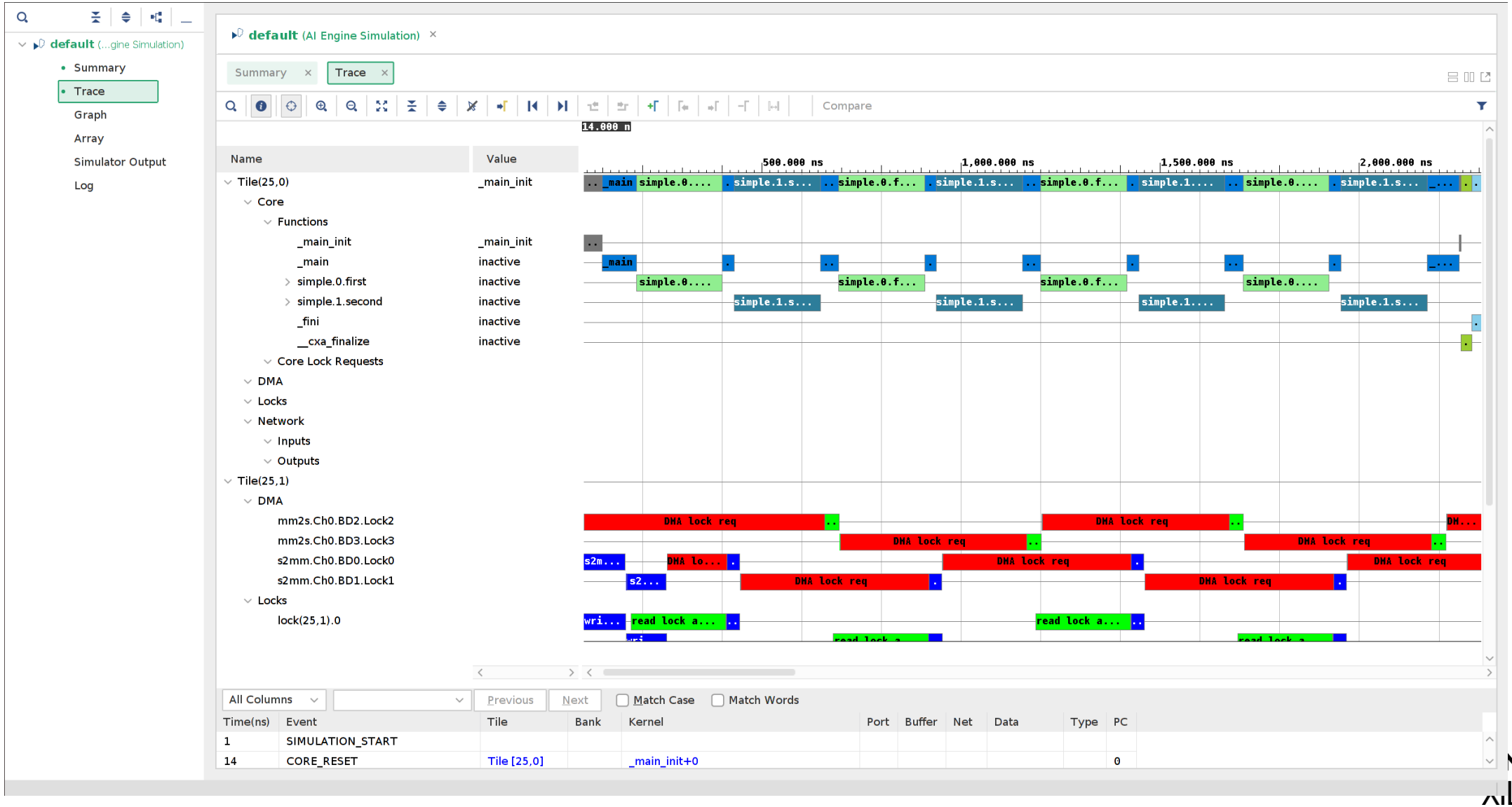


AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

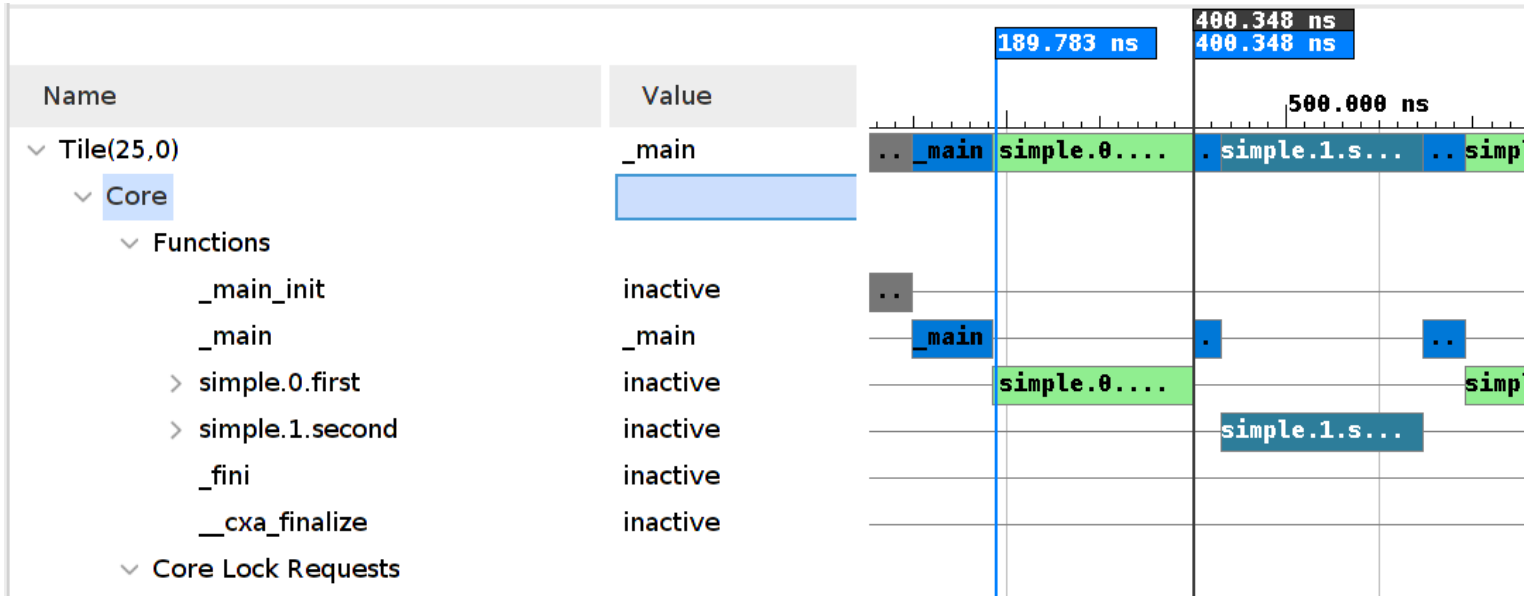
Vitis Analyzer - Trace



Starting out with the AI Engine tools

XILINX VITIS

Vitis Analyzer



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

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

AMD XILINX

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 ***

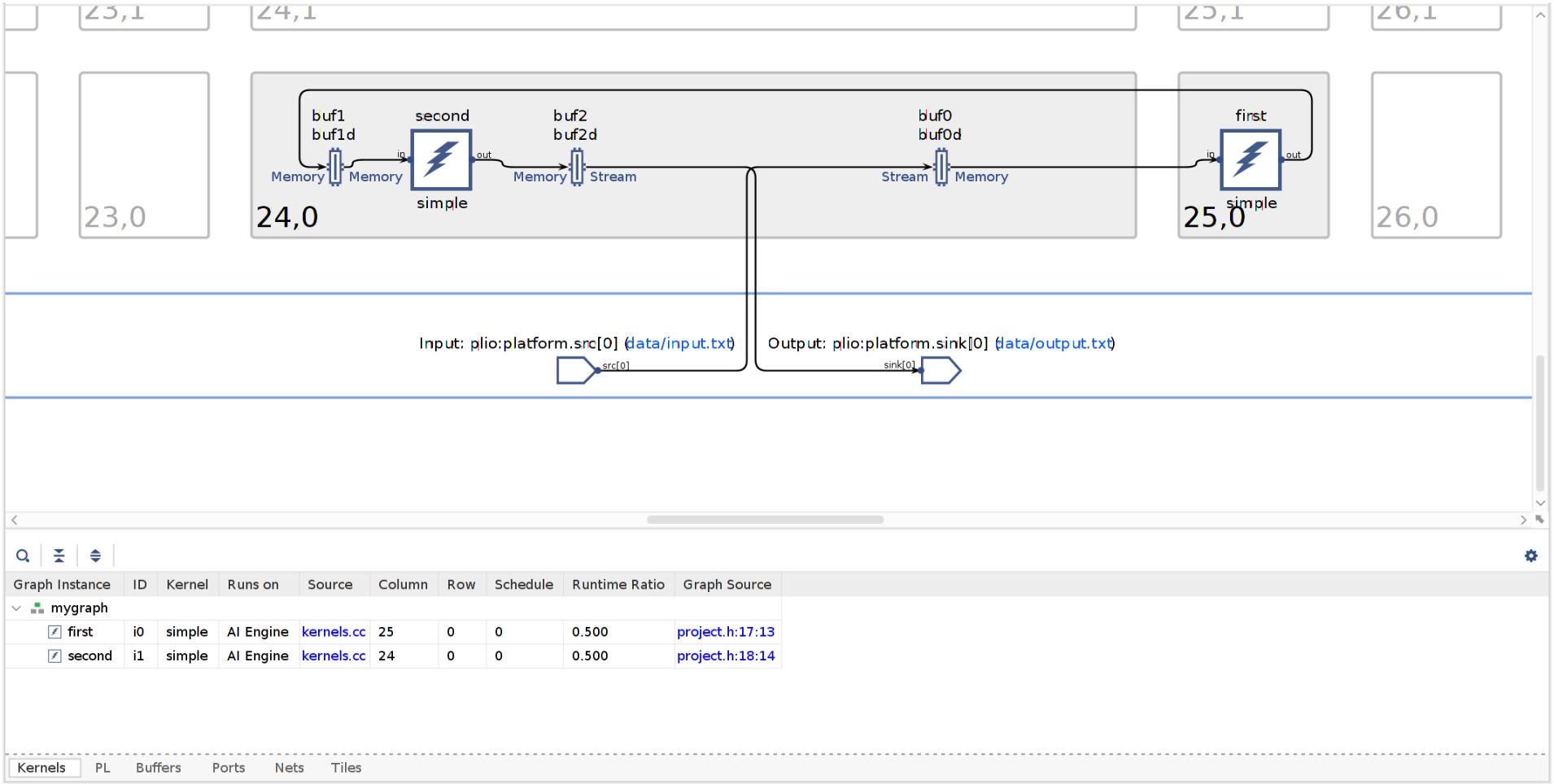
Starting out with the AI Engine tools

XILINX VITIS

Vitis Analyzer

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

Modify ratio number to 0.5



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	0.500	project.h:17:13
second	i1	simple	AI Engine	kernels.cc	24	0	0	0.500	project.h:18:14

Kernels | PL | Buffers | Ports | Nets | Tiles

AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Output File Timing Stamp

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

Modify ratio number to 0.5

project.h | project.cpp | kernels.cc | output.txt

```
10 8
T 774400 ps
14 12
T 777600 ps
18 16
T 780800 ps
22 20
T 784 ns
26 24
T 787200 ps
TLAST
30 28
T 963200 ps
0 2
T 966400 ps
4 6
T 969600 ps
8 10
T 972800 ps
12 14
T 976 ns
16 18
T 979200 ps
20 22
T 982400 ps
24 26
T 985600 ps
28 30
T 988800 ps
2 0
T 992 ns
6 4
```

output2.txt

```
10 8
T 761600 ps
14 12
T 764800 ps
18 16
T 768 ns
22 20
T 771200 ps
26 24
T 774400 ps
TLAST
30 28
T 1184 ns
0 2
T 1187200 ps
4 6
T 1190400 ps
8 10
T 1193600 ps
12 14
T 1196800 ps
16 18
T 1200 ns
20 22
T 1203200 ps
24 26
T 1206400 ps
28 30
T 1209600 ps
2 0
T 1212800 ps
6 4
```


AMD XILINX

Starting out with the AI Engine tools

XILINX VITIS

Output File Timing Stamp

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

Modify ratio number to 0.5

project.h | project.cpp | kernels.cc | output.txt

```
10 8
T 1049600 ps
14 12
T 1052800 ps
18 16
T 1056 ns
22 20
T 1059200 ps
26 24
T 1062400 ps
TLAST
30 28
T 1241600 ps
0 2
T 1244800 ps
4 6
T 1248 ns
8 10
T 1251200 ps
12 14
T 1254400 ps
16 18
T 1257600 ps
20 22
T 1260800 ps
24 26
T 1264 ns
28 30
T 1267200 ps
2 0
T 1270400 ps
6 4
```

output2.txt

```
10 8
T 1270400 ps
14 12
T 1273600 ps
18 16
T 1276800 ps
22 20
T 1280 ns
26 24
T 1283200 ps
TLAST
30 28
T 1692800 ps
0 2
T 1696 ns
4 6
T 1699200 ps
8 10
T 1702400 ps
12 14
T 1705600 ps
16 18
T 1708800 ps
20 22
T 1712 ns
24 26
T 1715200 ps
28 30
T 1718400 ps
2 0
T 1721600 ps
6 4
```


AMD XILINX

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


...
}
```


AMD XILINX

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

AMD
XILINX

APPENDIX: ISSUE

asm/errno.h: No such file or directory

Solution:

ln -s /usr/include/asm-generic /usr/include/asm

ERROR: [aiecompiler 77-753]

Solution:

Get AIE License from Xilinx website

Licensing Solution Center (xilinx.com)