



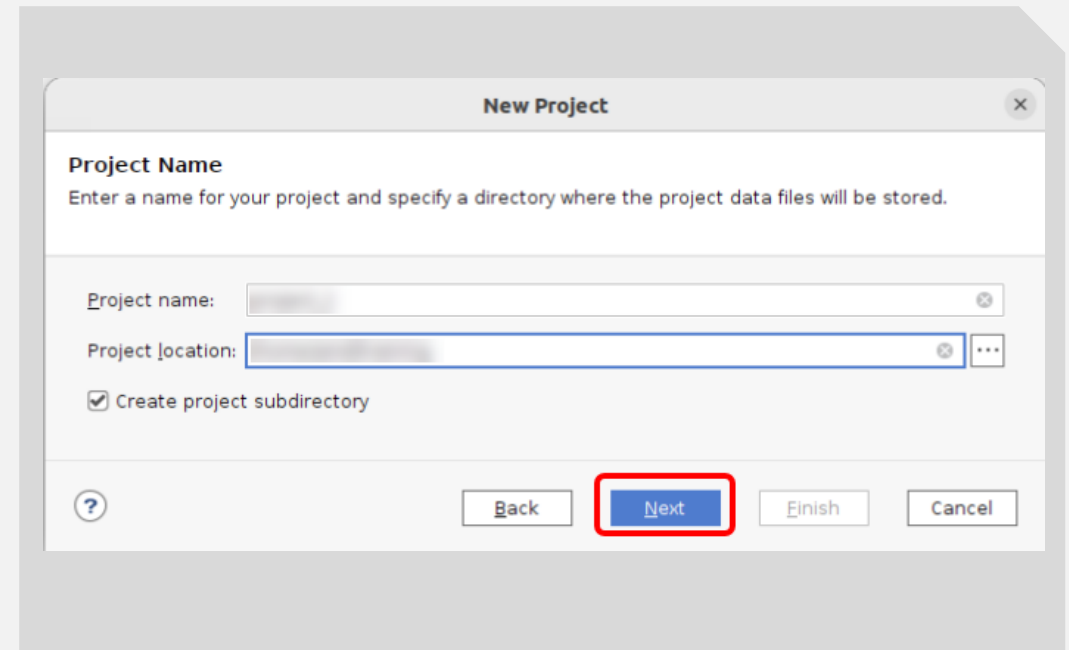
AMD Versal™ AI Engines for DSP End-to-End Design Flow

Agenda

-
1. [Creating a Vivado Extensible Platform](#)
 2. Creating an AI Engine Component in Vitis™ Unified IDE
 3. Reviewing the Source Files
 4. Configuring DSP Library Parameters Using .csv File
 5. Running AI Engine Compiler and Simulator, Vitis Analyzer to Measure Latency and Throughput
 6. Running Export to Vivado™ Tool Flow

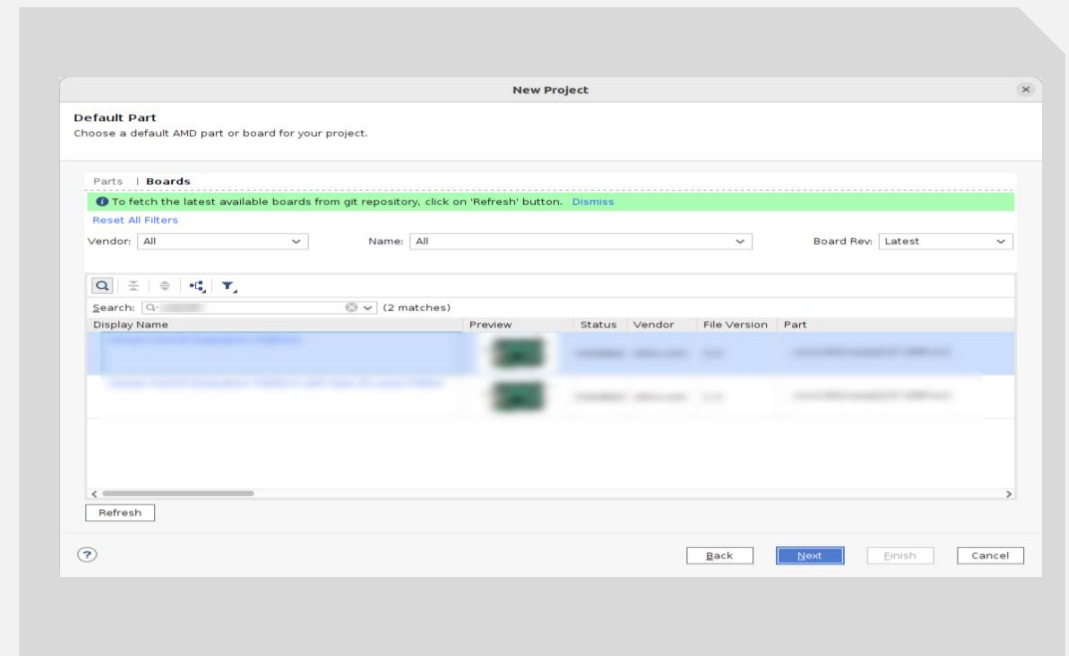
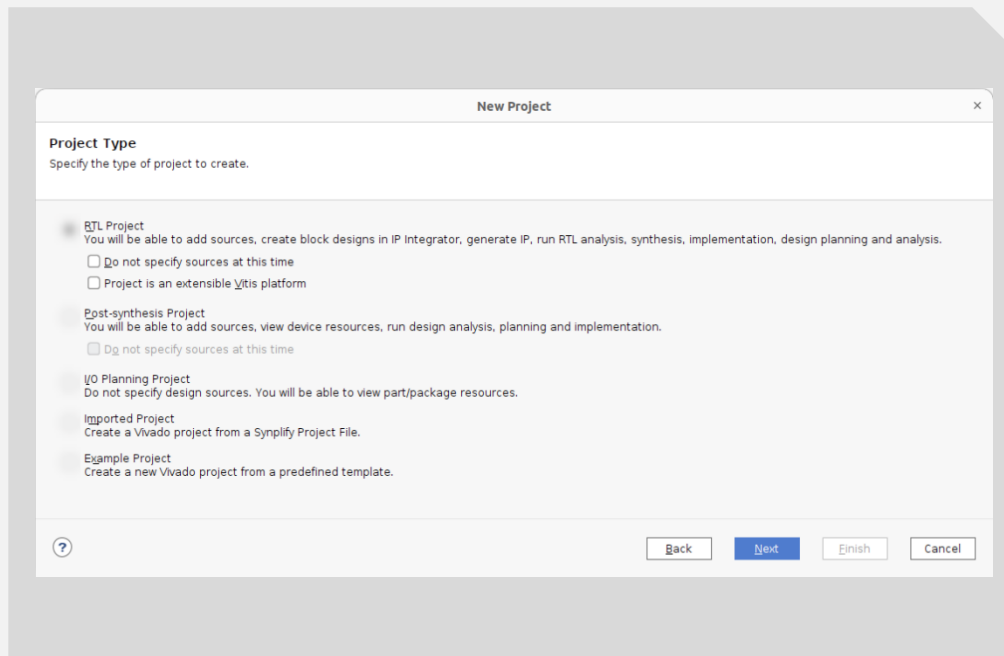
Creating a Vivado Extensible Platform

- Click the Vivado Design Suite icon () from the taskbar
- Click **Create Project** and click Next on New Project window, enter project details and click on Next



Creating a Vivado Extensible Platform


- Select the required **Project Type** and click on Next
- Now Add Sources and Constraints if required
- Choose a board required for the project and click Next
- Review the project summary and click on Finish

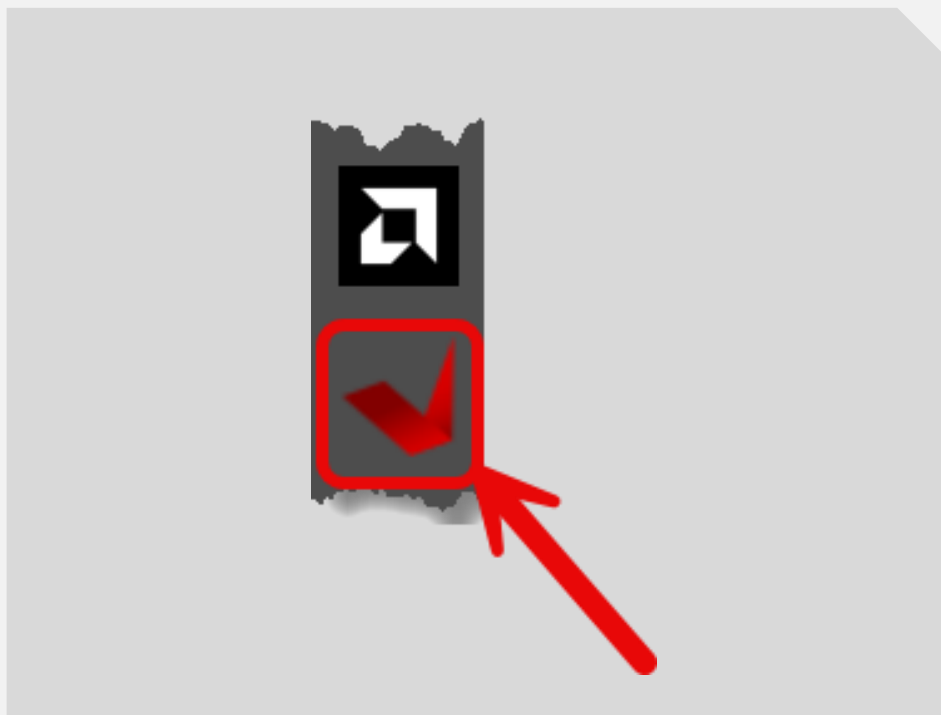


Agenda

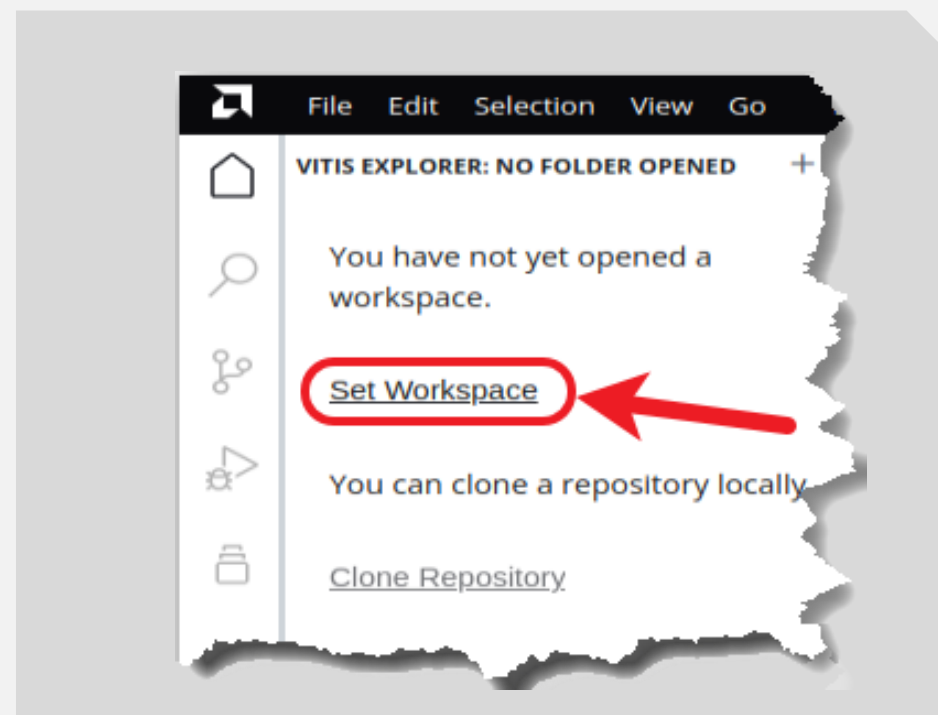
-
1. Creating a Vivado Extensible Platform
 2. Creating an AI Engine Component in Vitis™ Unified IDE
 3. Reviewing the Source Files
 4. Configuring DSP Library Parameters Using .csv File
 5. Running AI Engine Compiler and Simulator, Vitis Analyzer to Measure Latency and Throughput
 6. Running Export to Vivado™ Tool Flow

Creating an AMD Vitis Unified IDE Project

- Click the Vitis™ Unified IDE icon () from the taskbar



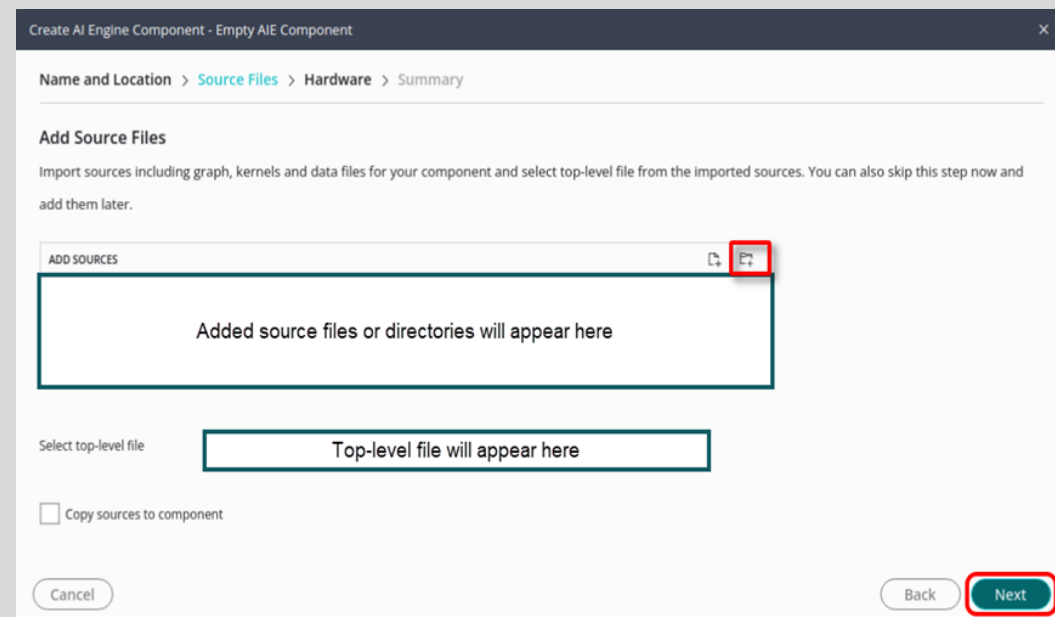
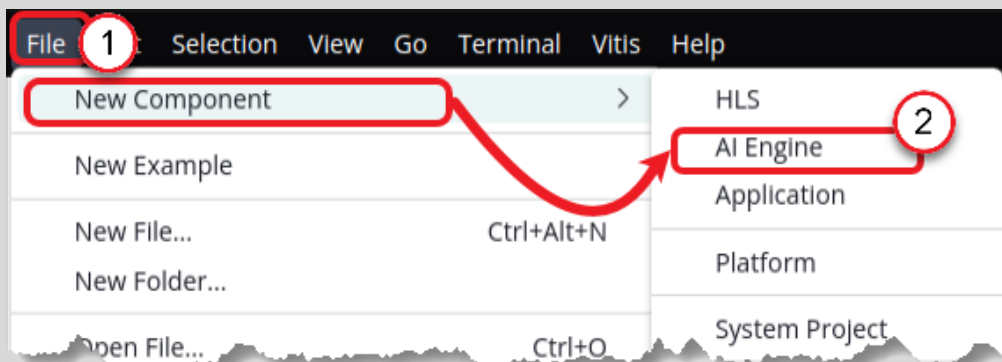
- Click **Set Workspace** to choose a workspace directory
- Click Open to proceed



Creating an AI Engine Component

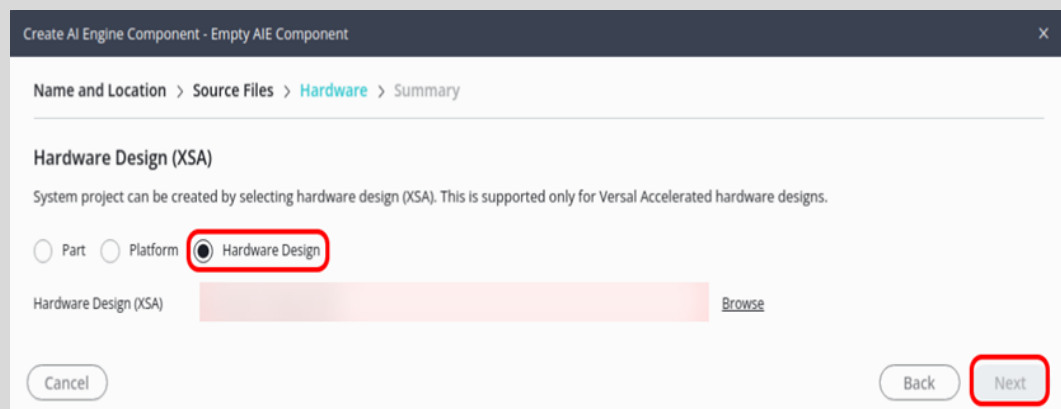
- File > New Component > AI Engine
- Component name (e.g., filter_design_acc)
- Choose Component location (default is workspace)

- Click Add Folders under Import Sources
- Select data and src directories

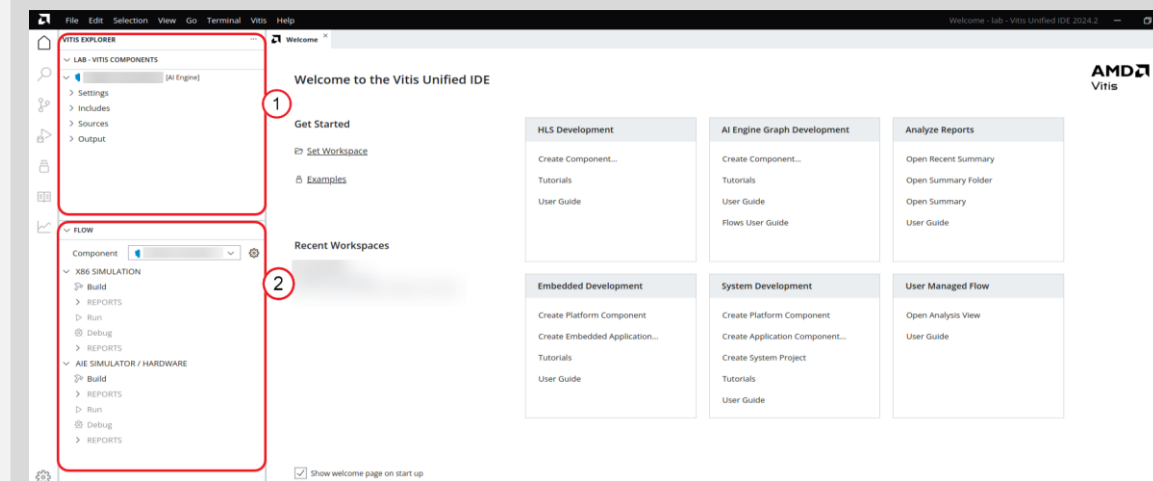


Creating an AI Engine Component

- Choose the required XSA
- Click Next



- Review Summary
- Click Finish
- Component appears in Components Window



Agenda

-
1. Creating a Vivado Extensible Platform
 2. Creating an AI Engine Component in Vitis™ Unified IDE
 3. [Reviewing the Source Files](#)
 4. Configuring DSP Library Parameters Using .csv File
 5. Running AI Engine Compiler and Simulator, Vitis Analyzer to Measure Latency and Throughput
 6. Running Export to Vivado™ Tool Flow

Reviewing the Source Files

- Open graph code
 - Path: [AI Engine] > Sources > src > fir1_graph.h
- For the FIR filter, the following parameters are declared:
 - TT_DATA: Data type
 - TT_COEFF: Coefficient type
 - TP_FIR_LEN: Length of the filter (overall length including zeros)
 - TP_SHIFT: Shifting value operated just before sending the data to the output
 - TP_RND: Truncation, rounding, rounding up, etc.
 - TP_INPUT_WINDOW_VSIZE: Size of the window (buffer) in samples
 - TP_CASC_LEN: Number of cascaded kernels to use for the FIR
 - TP_DUAL_IP: Use dual inputs if set to 1
 - TP_USE_COEFF_RELOAD: No reload if set to 0
 - TP_NUM_OUTPUTS: Number of ports to broadcast the output
 - TP_API: Set for windows (I/O buffers) or stream data APIs
 - TP_SSR: Scale throughput using parallelization

```
#pragma once

#include <adf.h>
#include <fir_sr_sym_graph.hpp>
#include <fft_ifft_dit_lch_graph.hpp>
#include "fir1_coeff.h"

using namespace adf;

using namespace xf::dsp::aie::fir::sr_sym;
using namespace xf::dsp::aie::fft::dit_lch;

// -----
// FIR Namespace
// -----

namespace fir1 {
    static constexpr int WIN_SIZE = 2048;
    typedef cint16 TT_DATA;
    typedef int16 TT_COEFF;
    static constexpr int TP_FIR_LEN = 2048; // Total number of taps
    static constexpr int TP_SHIFT = 0; // Depends on FXP properties
    static constexpr int TP_RND = 0; // how to set this
    static constexpr int TP_INPUT_WINDOW_VSIZE = 2048;
    static constexpr int TP_CASC_LEN = 1; // # of tiles in a cascade
    static constexpr int TP_DUAL_IP = 0; // Only used in stream mode (TP API = 1)
    static constexpr int TP_USE_COEFF_RELOAD = 0; // Set to 1 to use coefficient reload feature
    static constexpr int TP_NUM_OUTPUTS = 2; // Add optional 2nd output port for windows
    static constexpr int TP_API = 1; // Set to 1 for stream mode (windows otherwise)
    static constexpr int TP_SSR = 1; // Scale throughput using parallelization
};

// -----
// FFT Namespace
// -----

namespace fft1 {
    typedef cint16 TT_TYPE;
    typedef cint16 TT_TWIDDLE;
    static constexpr int TP_POINT_SIZE = 2048;
    static constexpr int TP_FFT_NIFFT = 2048;
    static constexpr int TP_SHIFT = 0; // Excludes twiddle shift
    static constexpr int TP_CASC_LEN = 1;
    static constexpr int TP_DYN_PT_SIZE = 2048;
    static constexpr int TP_WINDOW_SIZE = 2048;
    static constexpr int TP_API = 1; // Windows
    static constexpr int TP_PARALLEL_POWER = 1;
};
```

FIR filter
parameters

FFT design
parameters

Reviewing the Source Files

For the FFT, the following parameters are declared:

- TT_TYPE: Type of individual data samples input to and output from the transform function
- TT_TWIDDLE: Twiddle factors of the transform
- TP_POINT_SIZE: Number of samples processed by the FFT
- TP_FFT_NIFFT: To select the transform to perform (0 for IFFT and 1 for FFT)
- TP_SHIFT: Number of bits to shift accumulate down by before output
- TP_CASC_LEN: Number of bits to shift accumulate down by before output
- TP_DYN_PT_SIZE: Number of bits to shift accumulate down by before output
- TP_WINDOW_SIZE: Number of samples in the input window
- TP_API: Set for windows (I/O buffers) or stream data APIs
- TP_PARALLEL_POWER: Selects the parallelism factor as a power of 2. Values range from 0 to 4

```
#pragma once

#include <adf.h>
#include <fir_sr_sym_graph.hpp>
#include <fft_ifft_dit_lch_graph.hpp>
#include "fir1_coeff.h"

using namespace adf;

using namespace xf::dsp::aie::fir::sr_sym;
using namespace xf::dsp::aie::fft::dit_lch;

// -----
// FIR Namespace
// -----

namespace fir1 {
    static constexpr int WIN_SIZE = 2048;
    typedef cint16
    typedef int16
    static constexpr int TP_FIR_LEN = TT_DATA;
    static constexpr int TP_SHIFT = TT_COEFF;
    static constexpr int TP_RND = // Total number of taps
    static constexpr int TP_INPUT_WINDOW_VSIZE = // Depends on FXP properties
    static constexpr int TP_CASC_LEN = // how to set this
    static constexpr int TP_DUAL_IP = ;
    static constexpr int TP_USE_COEFF_RELOAD = // # of tiles in a cascade
    static constexpr int TP_NUM_OUTPUTS = // Only used in stream mode (TP API = 1)
    static constexpr int TP_API = // Set to 1 to use coefficient reload feature
    static constexpr int TP_SSR = // Add optional 2nd output port for windows
    // Set to 1 for stream mode (windows otherwise)
    // Scale throughput using parallelization
};

// -----
// FFT Namespace
// -----

namespace fft1 {
    typedef cint16
    typedef cint16
    static constexpr int TP_POINT_SIZE = TT_TYPE;
    static constexpr int TP_FFT_NIFFT = TT_TWIDDLE;
    static constexpr int TP_SHIFT = ;
    static constexpr int TP_CASC_LEN = // Excludes twiddle shift
    static constexpr int TP_DYN_PT_SIZE = ;
    static constexpr int TP_WINDOW_SIZE = ;
    static constexpr int TP_API = // Windows
    static constexpr int TP_PARALLEL_POWER = ;
};
```

FIR filter
parameters

FFT design
parameters

Reviewing the Source Files

- Review the other components
- Update the kernel connections in fir1_graph.h as per your design
 - filt_i.out [0] > fir_dut.in [0]
 - fir_dut.out [0] > fft_dut.in [0]
 - fir_dut.out [0] > filt_o.in [0]
 - fft_dut.out [0] > fft_o.in [0]

```

class fir1_graph : public graph {
public:

private:
    // Filter taps:
    std::vector<fir1::TT_COEFF> m_taps = std::vector<fir1::TT_COEFF>(FIR1_COEFF);

    // Filter class:
    using TT_FIR = fir_sr_sym_graph<fir1::TT_DATA,fir1::TT_COEFF,fir1::TP_FIR_LEN,fir1::TP_SHIFT,
                                   fir1::TP_RND,fir1::TP_INPUT_WINDOW_VSIZE,fir1::TP_CASC_LEN,
                                   fir1::TP_DUAL_IP,fir1::TP_USE_COEFF_RELOAD,fir1::TP_NUM_OUTPUTS,
                                   fir1::TP_API,fir1::TP_SSR>;

    // FFT class:
    using TT_FFT = fft_ifft_dit_1ch_graph<fft1::TT_TYPE,fft1::TT_TWIDDLE,fft1::TP_POINT_SIZE,
                                           fft1::TP_FFT_NIFFT,fft1::TP_SHIFT,fft1::TP_CASC_LEN,
                                           fft1::TP_DYN_PT_SIZE,fft1::TP_WINDOW_SIZE,fft1::TP_API,
                                           fft1::TP_PARALLEL_POWER>;

public:
    input_plio filt_i; // top level input
    output_plio filt_o; // top level output
    output_plio fft_o; // fft output

    TT_FIR fir_dut; // for fir function
    TT_FFT fft_dut; // for fft function

    fir1_graph(void) : fir_dut( m_taps ), fft_dut()
    {
        filt_i = input_plio::create("PLIO_fir_i",plio_64_bits,"data/sig_i.txt");
        filt_o = output_plio::create("PLIO_fir_o",plio_64_bits,"data/fir_o.txt");
        fft_o = output_plio::create("PLIO_fft_o",plio_64_bits,"data/fft_o.txt");
        connect<>(      ); //connect the top level input to fir
        connect<>(      ); //connect the fir output to fft input
        connect<>(      ); //connect the fir output to top level fir output
        connect<>(      ); //connect the fft output to top level fft output
    }
}

```

Kernel coefficient

FIR filter declaration

FFT declaration

Filter initialization with I/O data and graph connections

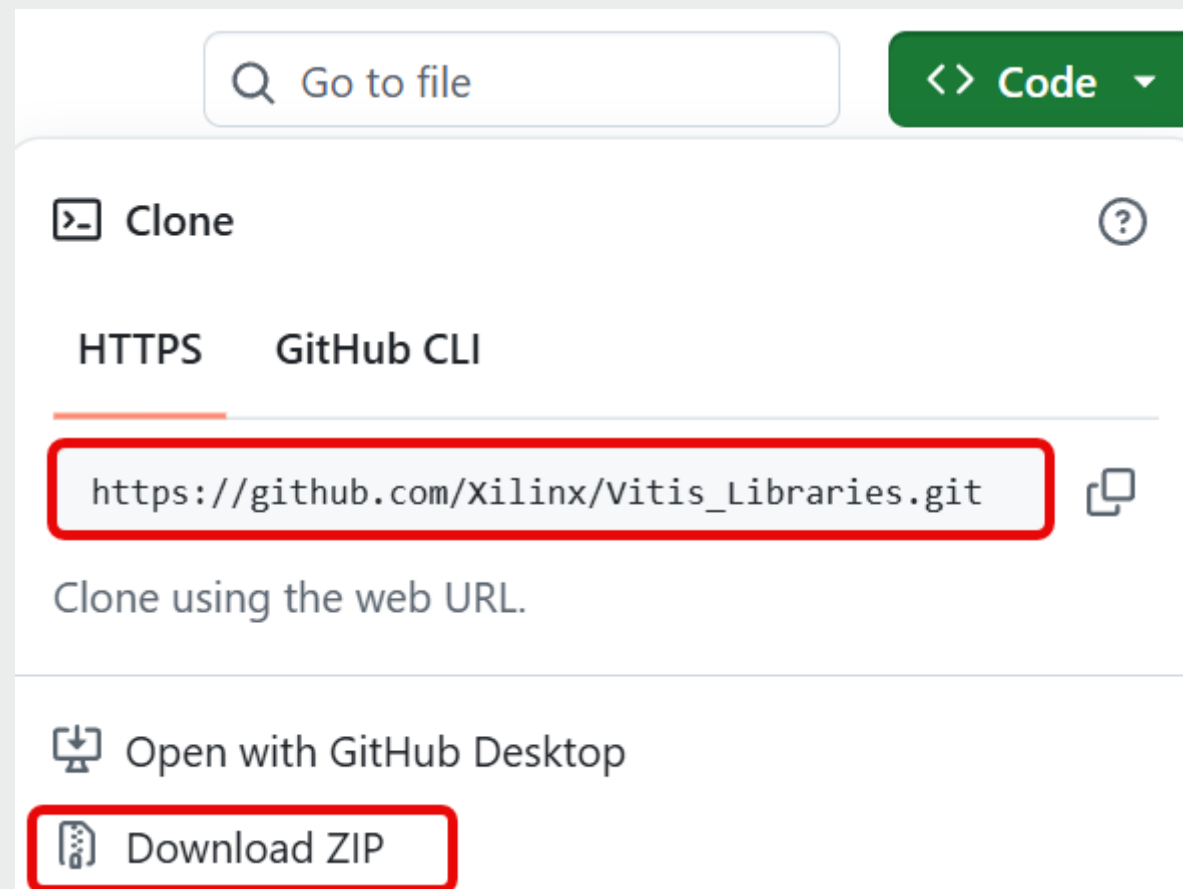
Agenda

-
1. Creating a Vivado Extensible Platform
 2. Creating an AI Engine Component in Vitis™ Unified IDE
 3. Reviewing the Source Files
 4. **Configuring DSP Library Parameters Using .csv File**
 5. Running AI Engine Compiler and Simulator, Vitis Analyzer to Measure Latency and Throughput
 6. Running Export to Vivado™ Tool Flow

How to Download / Import Libraries from GitHub

AMD Vitis™ DSP Library

https://github.com/Xilinx/Vitis_Libraries



Configuration Parameters

- Different set of parameters for different functions
- Available in [AMD Vitis™ Libraries Configuration Parameters](#)

The screenshot shows the 'Vitis Libraries' interface with the '2025.1 English' version selected. The left sidebar lists various configuration categories, with 'FFT Window Configuration Parameters' highlighted. The main content area displays the 'FFT Window Configuration Parameters' table, which is outlined in red. The table lists parameters such as DATA_TYPE, COEFF_TYPE, POINT_SIZE, SHIFT, WINDOW_VSIZE, DYN_PT_SIZE, API_IO, and WINDOW_CHOICE, along with their types, default values, and descriptions.

FFT Window Configuration Parameters

For the FFT Window library element, use the following list of configurable parameters and default values.

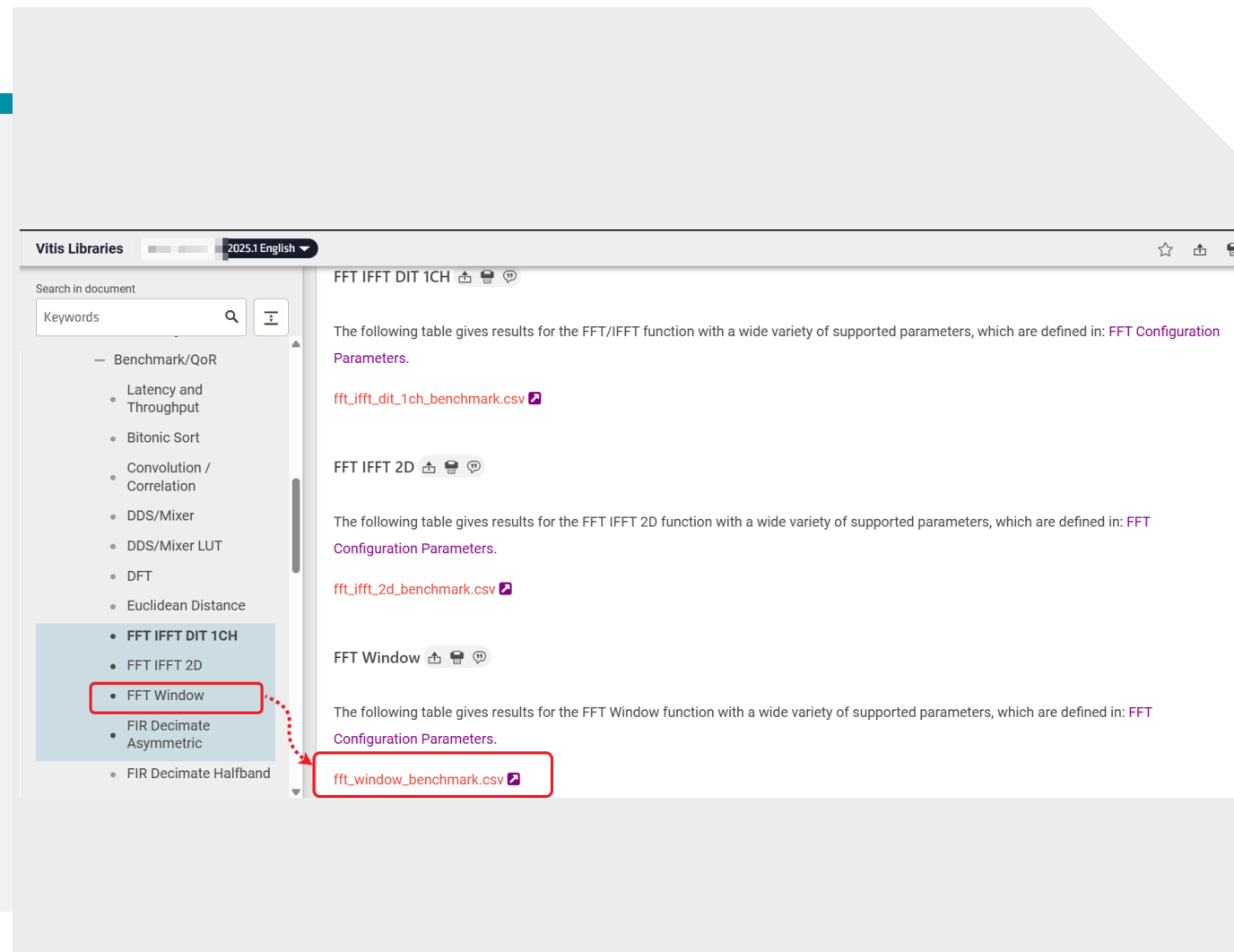
Table 97 FFT Window Configuration Parameters

Name	Type	Default	Description
DATA_TYPE	typename	cint16	Data Type.
COEFF_TYPE	typename	cint16	Coeff Type.
POINT_SIZE	unsigned	1024	FFT point size.
SHIFT	unsigned	17	See Common Configuration Parameters
WINDOW_VSIZE	unsigned	1024	Input/Output window size. By default, set to: \$(POINT_SIZE).
DYN_PT_SIZE	unsigned	0	Enable (1) Dynamic Point size feature.
API_IO	unsigned	0	Graph's port API. 0: window 1: stream
WINDOW_CHOICE	unsigned	0	Supported types: 0: Hamming 1: Hann 2: Blackman

Configuration Databases

AMD Vitis™ DSP Library – Configuration Database

- Gives an early estimate of power, latency, throughput, and resource utilization
- Benchmarks are for different combination of Library parameters such as datatypes and AI Engine types
- Has ~3K to 5K test results
- Is accessed as a CSV file located at:
[Vitis_Libraries/dsp/docs/src/csv_data_files/L2](https://github.com/Xilinx/Vitis_Libraries/tree/master/dsp/docs/src/csv_data_files/L2) at 2025.1 · Xilinx/Vitis_Libraries · GitHub



Filtering and Sorting CSV table

FFT Window Function – Filtering the values in the CSV table using the Excel sheet in Desktop

2025.1

Vitis_Libraries / dsp / docs / src / csv_data_files / L2 / fft_window_benchmark.csv

Go to file

2 people and GitHub Enterprise

Squashed 'dsp' changes from 118e8f053...3f6ed094... b5868f2 · 2 months ago

3782 lines (3782 loc) · 340 KB

Preview Code Blame

Search this file

	Library Element	AIE_VARIANT	TT_DATA	TT_COEFF	TP_POINT_SIZE	TP_WINDOW_VSIZE	TP_DYN_PT_SIZE	TP_SSR	TI
1	fft_window	AIE	cfloat	float	256	512	0	2	0
2	fft_window	AIE	cfloat	float	256	1024	0	2	0

File Home Insert Draw Page Layout Formulas Data Review View Automate Help Acrobat

1

Get Data From Text/CSV From Picture From Web Recent Sources From Table/Range Existing Connections

Refresh All

Queries & Connections Properties Workbook Links

Organization Stocks Currencies Geography

Sort & Filter Filter

2

Clear Reapply Advanced

A1

Library Element

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Library Element	AIE_VARIANT	TT_DATA	TT_COEFF	TP_POINT_SIZE	TP_WINDOW_VSIZE	TP_DYN_PT_SIZE	TP_SSR	TI							
2	fft_window.A					512	0	2	0	0	979	1239	8	2	24652	2234 22
3	fft_window.A					1024	0	2	0	0	1754	1245	10	2	41036	2234 22
4	fft_window.A					4096	1	2	0	0	3410	2485	18	2	73614	2912 29
5	fft_window.A					2048	0	2	0	0	3391	1248	10	2	73804	2234 22
6	fft_window.A					4096	0	2	0	0	4008	1248	18	2	139340	2234 22
7	fft_window.A					512	0	2	0	0	947	1239	8	2	27724	2222 22
8	fft_window.A					1024	0	2	0	0	1790	1245	9	2	44108	2250 22
9	fft_window.A					2048	0	2	0	0	3390	1247	10	2	76876	2250 22
10	fft_window.A					4096	0	2	0	0	6669	1248	18	2	142412	2250 22
11	fft_window.A					1024	0	2	0	0	1752	1245	12	2	50252	2230 22
12	fft_window.A										3392	1248	12	2	83020	2258 22
13	fft_window.A										6674	1248	20	2	148556	2258 22
14	fft_window.A										613	303	8	2	9100	3134 31
15	fft_window.A										534	2381	9	2	19342	2832 28
16	fft_window.A										931	378	8	2	11148	3230 32
17	fft_window.A										1456	474	8	2	15244	3294 32
18	fft_window.A										2594	500	8	2	23436	3422 34
19	fft_window.A										4701	561	10	2	39820	3262 32
20	fft_window.A										645	1239	5	1	10759	230
21	fft_window.A										352	934	9	2	12684	3166 31
22	fft_window.A										528	1196	9	2	16780	3294 32
23	fft_window.A										959	1221	8	2	24972	3422 34
24	fft_window.A										1763	1235	10	2	41356	3262 32
25	fft_window.A										3402	1242	18	2	74124	3262 32
26	fft_window.A										531	1190	9	2	19852	3256 32
27	fft_window.A										955	2443	8	2	27534	3040 30
28	fft_window.A										962	1221	8	2	28044	3448 34
29	fft_window.A										1767	1236	10	2	44428	3288 32
30	fft_window.A										3404	1242	18	2	77196	3288 32
31	fft_window.A										950	1234	12	2	24198	2268 22

3

TP_POINT_SIZE

Sort Smallest to Largest Sort Largest to Smallest Sort by Color Sheet View Clear Filter From "TP_POINT_SIZE" Filter by Color Number Filters

Search

☒ (Select All) ☒ 16 ☒ 32 ☒ 64 ☒ 128 ☒ 256 ☒ 512 ☒ 1024 ☒ 4096

OK Cancel

Equals...

Does Not Equal...

Greater Than...

Greater Than Or Equal To...

Less Than...

Less Than Or Equal To...

Between...

Top 10...

Above Average

Below Average

Custom Filter...

AMD together we advance_

17

Agenda

-
1. Creating a Vivado Extensible Platform
 2. Creating an AI Engine Component in Vitis™ Unified IDE
 3. Reviewing the Source Files
 4. Configuring DSP Library Parameters Using .csv File
 5. Running AI Engine Compiler and Simulator, Vitis Analyzer to Measure Latency and Throughput
 6. Running Export to Vivado™ Tool Flow

Building and Simulating the Project

- Click Build under AIE Simulator/Hardware to build the component
- Add Launch Configuration
 - Path: Settings > launch.json
 - Select filter_design_acc_aiesim_1 config
- Click Run under AIE Simulator/Hardware
- Monitor TASK window

The image illustrates the process of building and simulating a project in Vitis, showing four key steps:

- Vitis Explorer:** The 'LAB - VITIS COMPONENTS' tree is shown. The 'Settings' folder is expanded, and 'launch.json' is selected (circled in red and labeled '2').
- launch.json:** The file is open in the editor, showing the configuration for '_aiesim_1' (circled in red).
- FLOW Pane:** The 'AIE SIMULATOR / HARDWARE' section is expanded, and the 'Run' button is highlighted (circled in red and labeled with a red arrow).
- OUTPUT Window:** The simulation output is displayed, showing the completion of the simulation and the release of the AIEsim feature license (circled in red and labeled with a red arrow).

Latency and Throughput Estimates

Average Throughput Computed

Displayed at End of AIE Simulation

VCD file generates latency and throughput estimates

```
aiesimulator -pkg-dir=./Work -dump-vcd foo -options-file=aiesim-options.txt
```

Continuous Latency

Between specific input and output ports

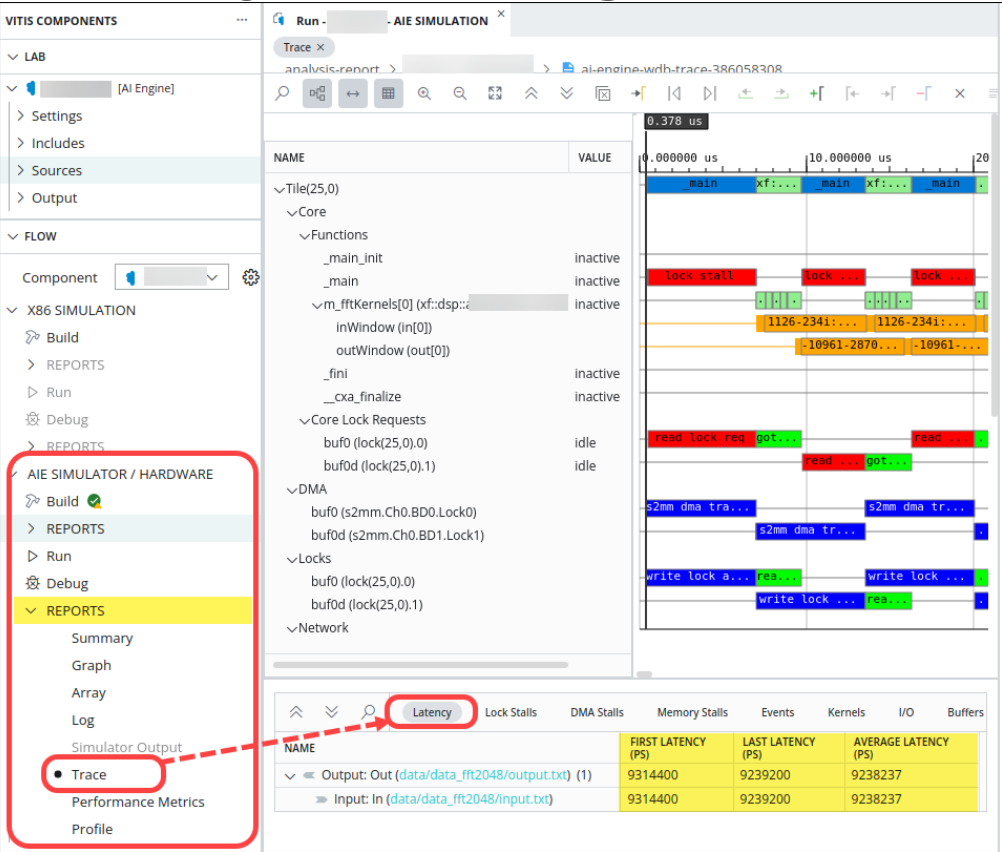
Continuous Throughput

At a specific input or output port

Vitis™ Unified IDE > AIE Simulation Reports > Trace > Opens Analysis View

Command line: `vitis_analyzer aie/aiesimulator_output/default.aierun_summary`

Latency & Throughput Table



First Latency	Last Latency	Average Latency
Latency time between first input to first output	Latency time between last input to last output	Difference between avg. output sample time and avg. input sample time

Plot or Export Continuous Latency and Throughput

	Latency	Lock Stalls	DMA Stalls	Events	Kernels	I/O	Buffers	Ports	Nets	Tiles
NAME	FIRST LATENCY (PS)	LAST LATENCY (PS)	AVERAGE LATENCY (PS)							
I/O Latency (1)										
Output: PLIO_Out0 (/data/soutput.txt) (1)	20549600	50151200	39885777							
Input: PLIO_In0 (/data/Sinput.txt)	20549600	50151200	39885777							
Kernel Latency (1)										
Sgraph (2)										
Kernel to Kernel Latency (1)										
Sgraph (1)										

	Latency	Lock Stalls	DMA Stalls	Memory Stalls	Kernels	I/O	Buffers	Ports	Nets	Tiles	Interface Channels	DMA Channels
NAME					FREQUENCY (MHZ)	THROUGHPUT (MBYTES/S)	BUFFERS	CONNECTED PORTS	COLUMN	CHANNEL ID	LOCATION CONSTRAINT	PACKET IDS
fft_design (2)												
Input: In (/data/data_fft2048/input.txt)	PLIO				312.5	1250.076299	2	1	25	0		
Output: Out (/data/data_fft2048/output.txt)	PLIO				312.5	1250.152607	2	1	25	0		

Agenda

-
1. Creating a Vivado Extensible Platform
 2. Creating an AI Engine Component in Vitis™ Unified IDE
 3. Reviewing the Source Files
 4. Configuring DSP Library Parameters Using .csv File
 5. Running AI Engine Compiler and Simulator, Vitis Analyzer to Measure Latency and Throughput
 6. Running Export to Vivado™ Tool Flow

AMD Vitis Export to Vivado Flow

Enables bi-directional hardware hand-offs between Vivado™ Design Suite and Vitis™ tools

```
v++ -link --export_archive
```

Create custom Vivado platform (Flat/BDC) using RTL, HLS, or IP catalog

Flow can be repeated for any number of design iterations

XSA file can be exported from Vivado IDE and passed back to Vitis tools

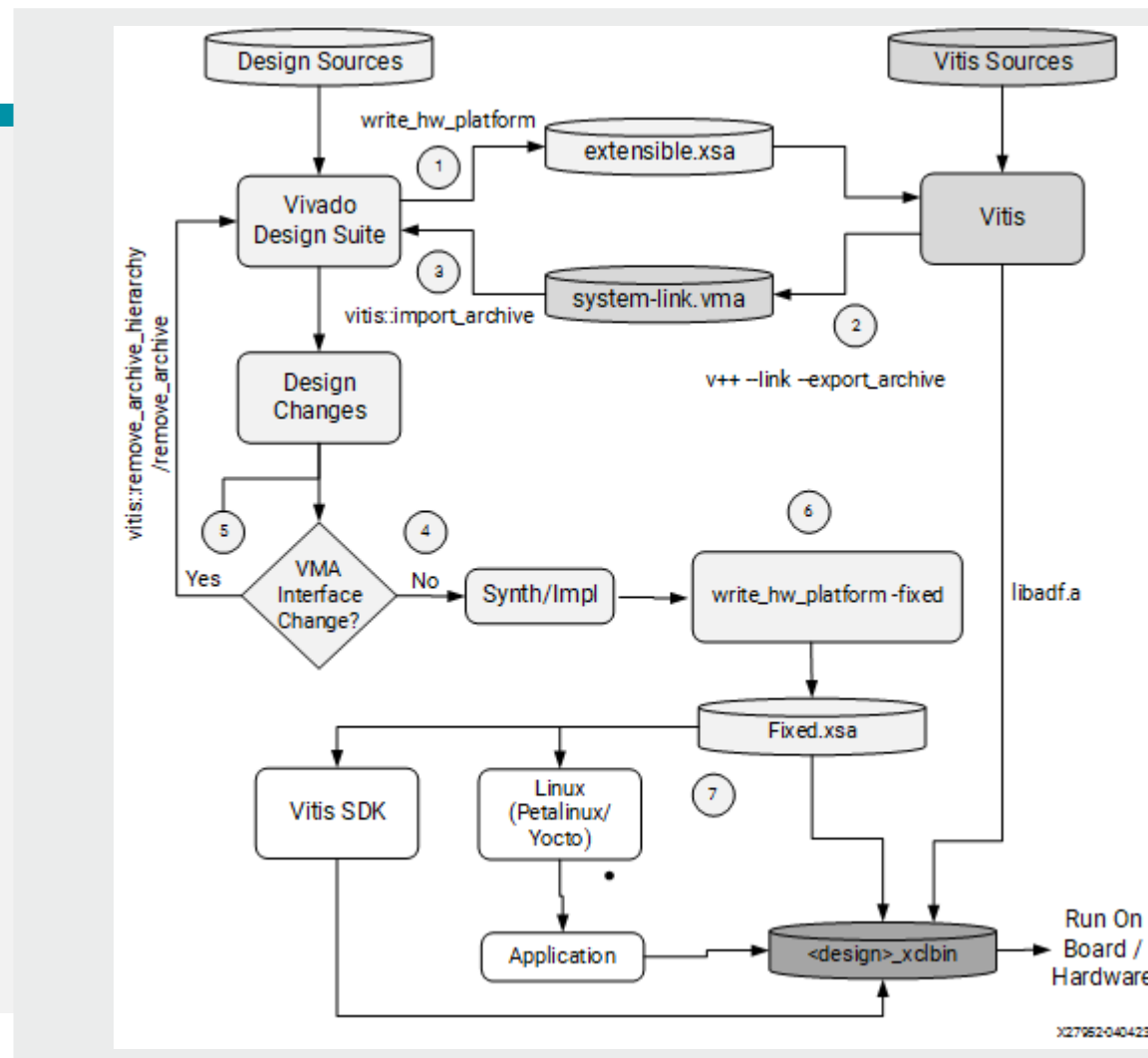
Updated VMA file can be reimported into Vivado IDE

v++ compiler operates on Vivado project that has been encapsulated in extensible XSA

Support for modifications to Vivado project that do not invalidate contract between imported design and XCLBIN

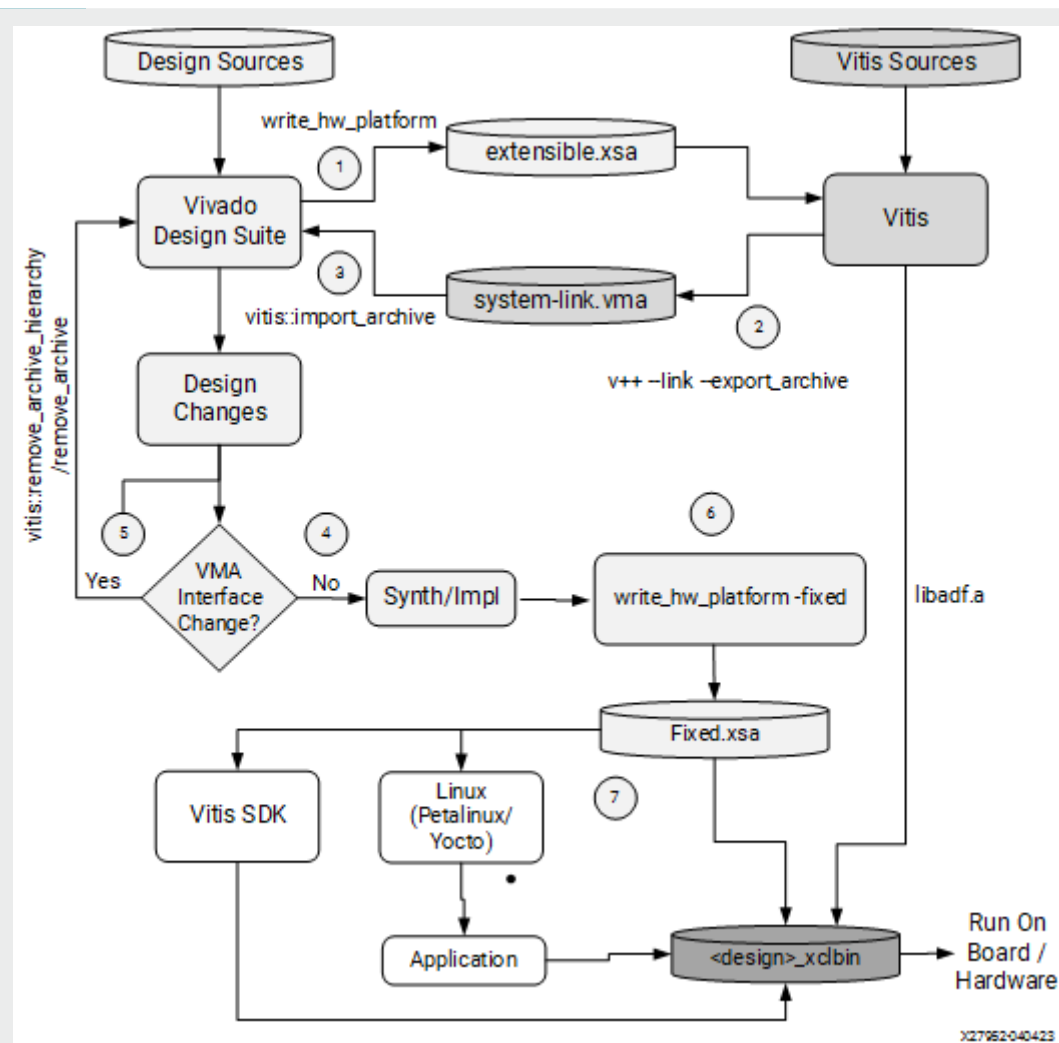
Vitis Export Flow Implementation

1. Import XSA in Vitis™ to compile and link:
 - AI Engine graph (libadf.a)
 - PL kernels (.xo)
 - Update system.cfg, run Vitis linker
2. Export Vitis Metadata Archive (VMA)
 - **`v++ --link --export_archive --platform <xsa> -
-config system.cfg <xo> libadf.a -o
<vma>.vma`**



AMD Vitis Export Flow Implementation

3. Import VMA into Vivado™ tools:
 - **vitis::import_archive ./<vma>.vma**
 - Creates Vitis region block design (read-only Vitis hierarchy)
4. Modify design in Vivado as needed
 - For PL/AIE updates → remove VMA (**vitis::remove_archive**)
 - Re-export XSA and repeat Vitis™ tool flow if required
5. After implementation, generate fixed XSA:
 - **write_hw_platform -fixed ./<fixed_xsa>.xsa**
6. Use fixed XSA for:
 - Yocto™ / Vitis Embedded apps
 - Bare-metal or hardware validation
7. For emulation:
 - Generate sim-included XSA (include_sim_content)
8. Package and create deployable .xclbin:
 - **v++ --package -t <hw|hw_emu> --xsa <fixed_xsa>**



In this demo, we'll walk you through the complete end-to-end flow of developing a Vitis Subsystem, or VSS, and integrating it with a custom Vivado extensible platform — targeting the AMD Versal VCK190 device.

Watch the video on YouTube - AMD Versal™ AI Engines for DSP End-to-End Design Flow

<https://youtu.be/J6NIZHCKqlg>

Summary

1

Source AMD Vitis™ tools and configure Linux® sysroot

2

Use Makefiles to compile AI Engine, HLS, RTL, and validate functionality

3

Configure core blocks and export XSA for Vitis integration

4

Link components in Vitis, generate VMA, and import back into the Vivado™ for final design

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, Vivado, 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. Yocto Project is a trademark of The Linux Foundation. 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.

