



Vitis™ System Integration Demo

A Step-to-Step Tutorial to Integrate RTL and HLS Kernels
for Accelerated Computing



Adaptive Computing Acceleration with Alveo



Computational Storage



Database and Data Analytics



Financial Technology



High Performance Computing



Network Acceleration



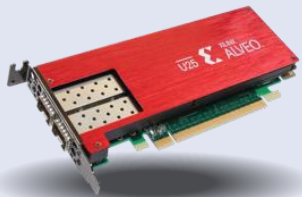
Video and Imaging



Machine Learning



Tools and Services



Alveo U25



Alveo U50



Alveo U200



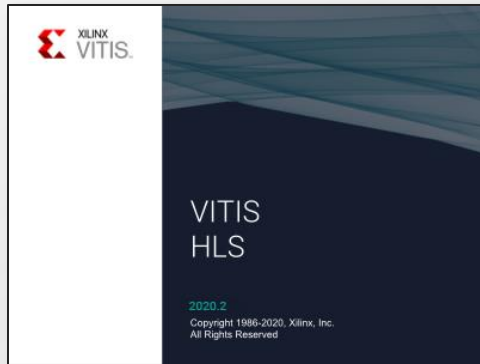
Alveo U250



Alveo U280



Inclusive Hardware Design Methodology



High Level Synthesis (HLS) Modules

Image
Processing

Matrix
Operation

Encryption

Data
Compression

Others..



RTL Modules

Video
CODEC

Image
CODEC

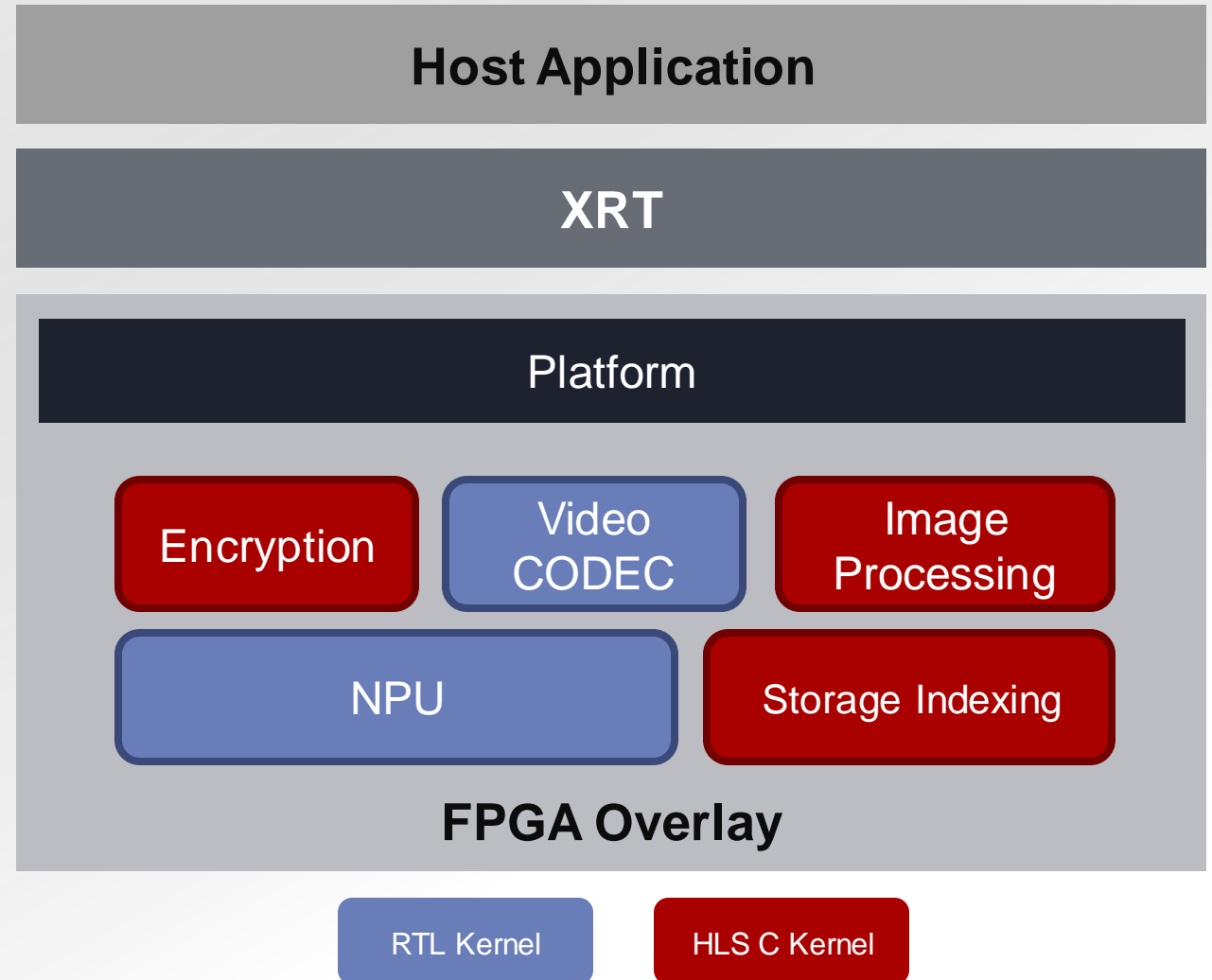
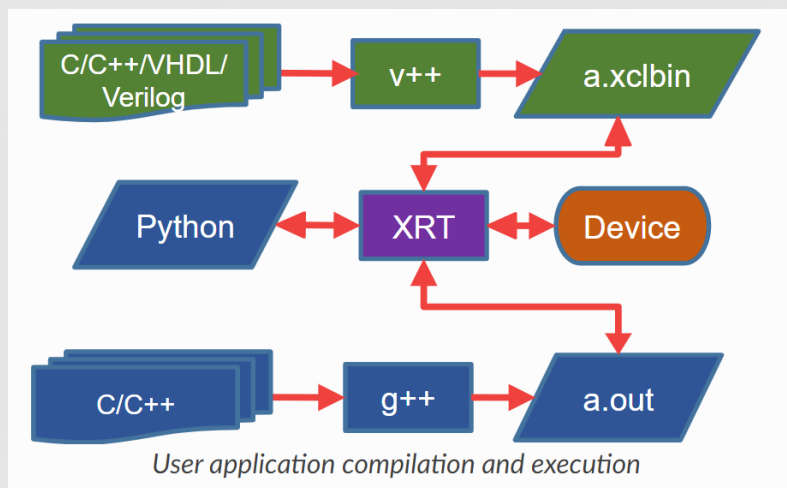
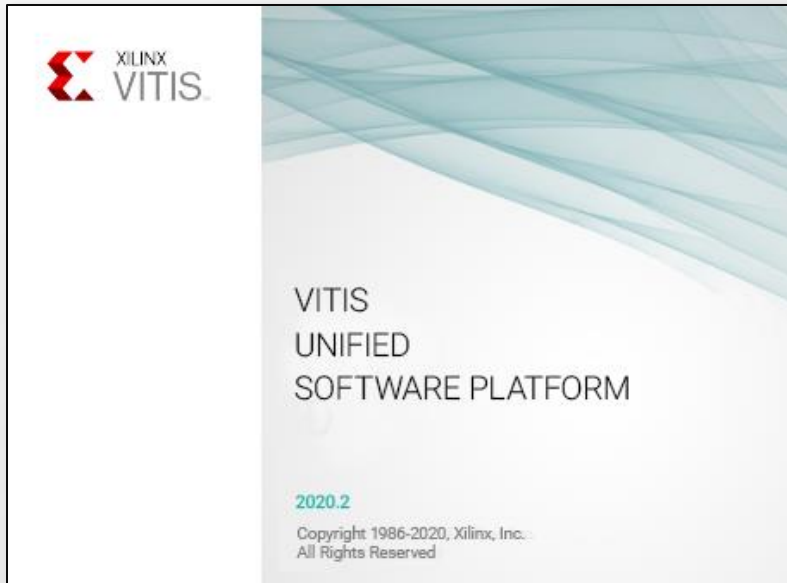
Processor

NPU

Others..



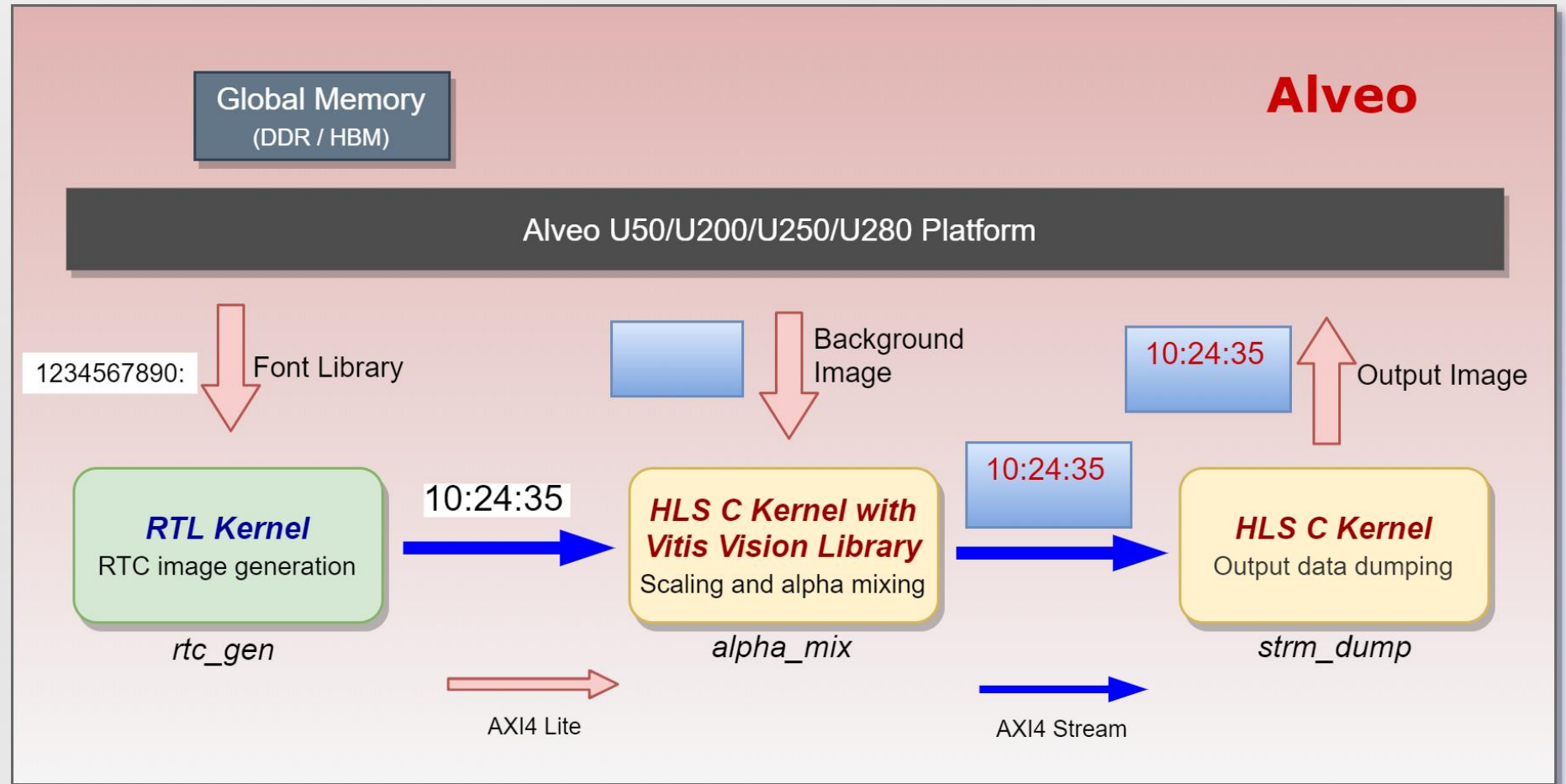
Fast Mixed Kernel Integration Flow with Vitis



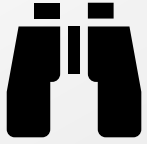
Mixed Kernel Design Example in Vitis Tutorials

Mixed HDL/HLS Kernel Integration Example

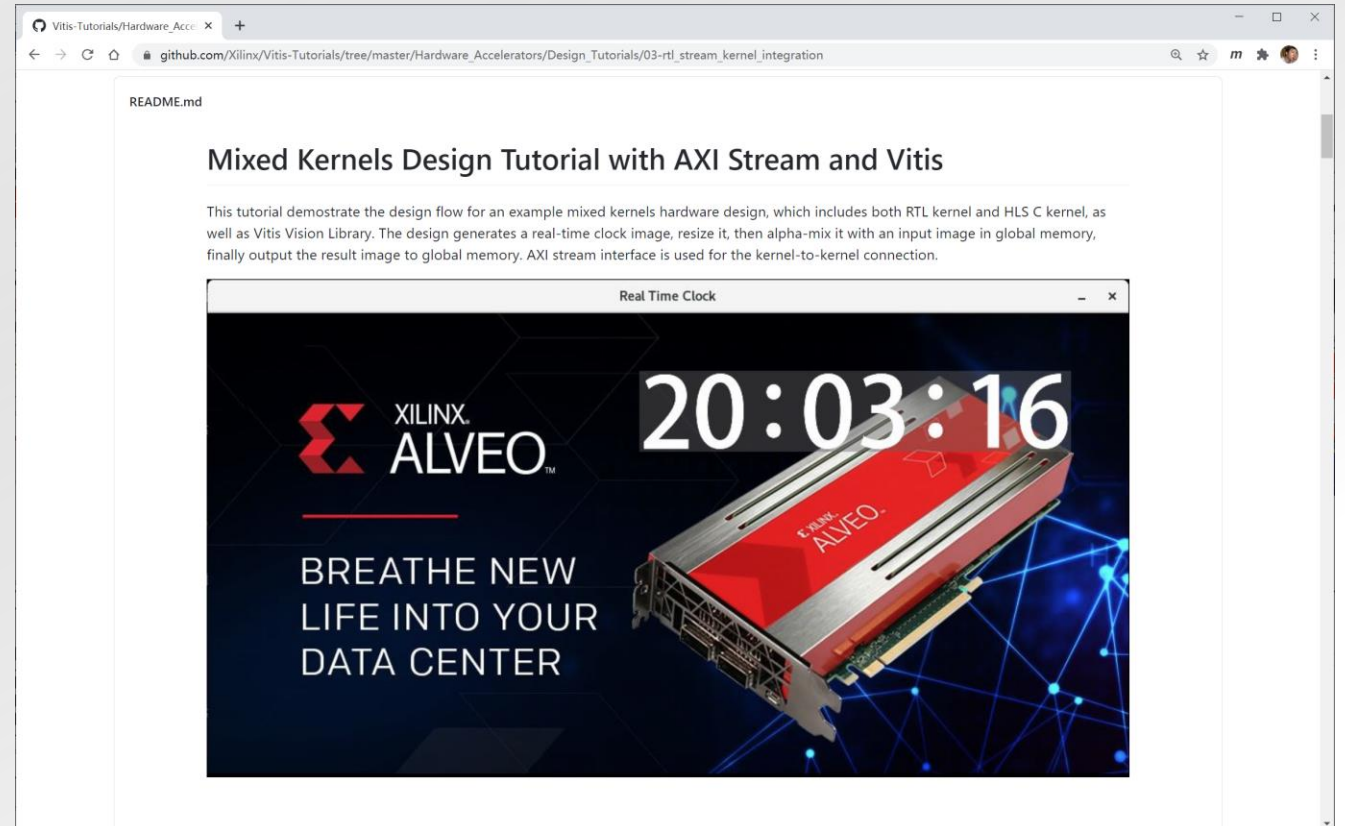
- rtc_gen*
 - ✓ Verilog Kernel
- alpha_mix*
 - ✓ HLS C Kernel with Vitis Vision Library
- strm_dump*
 - ✓ HLS C Kernel



Find the Example Design Resources



- ▶ Get Vitis™ Tutorials Repository
 - <https://github.com/Xilinx/Vitis-Tutorials>
- ▶ Position of This Example Design
 - Vitis-Tutorials
 - Hardware_Accelerators
 - Design_Tutorials
 - 03-rtl_stream_kernel_integration



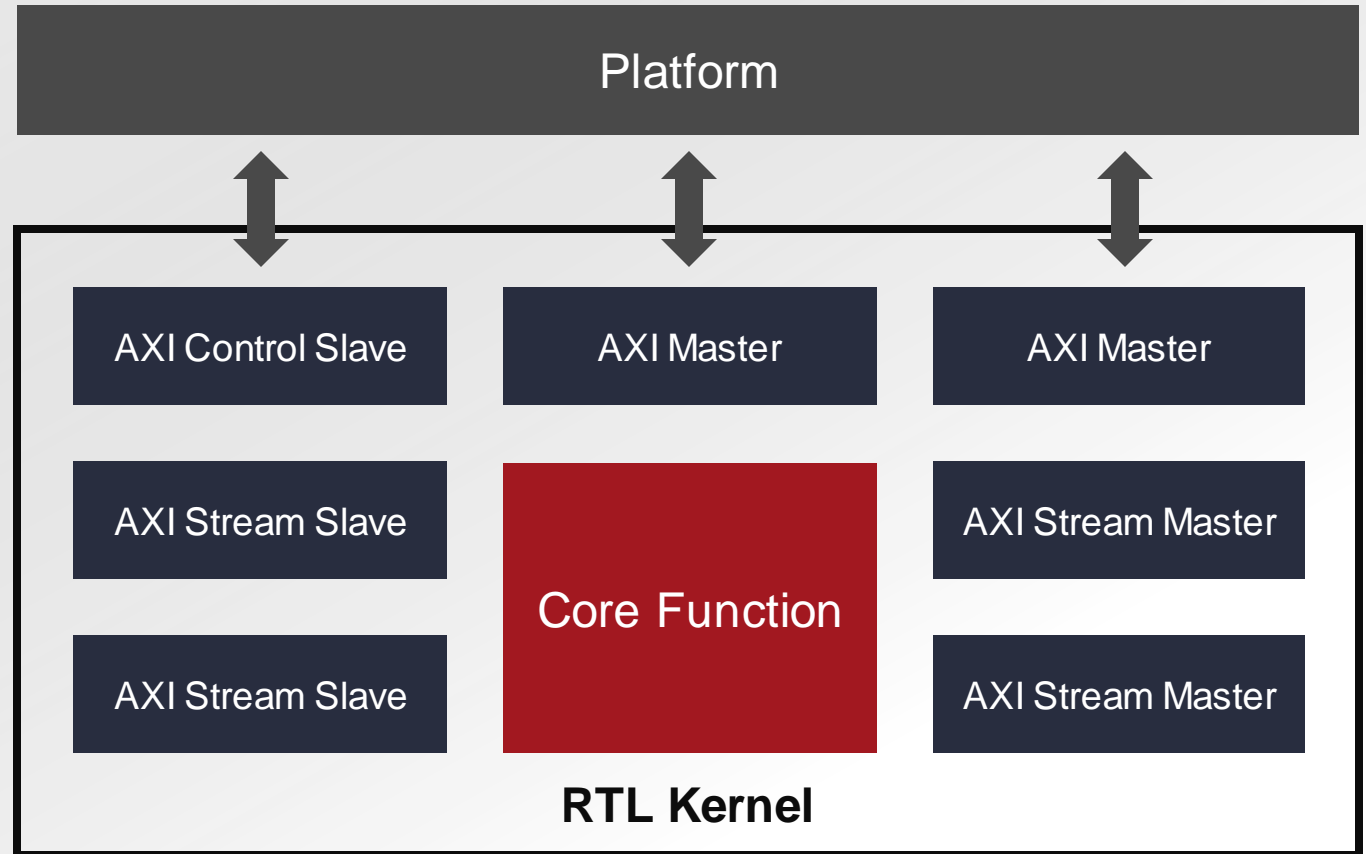
Linux git command lines to get the example design

```
git clone https://github.com/Xilinx/Vitis-Tutorials.git  
cd Vitis-Tutorials/Hardware_Accelerators/Design_Tutorials/03-rtl_stream_kernel_integration
```

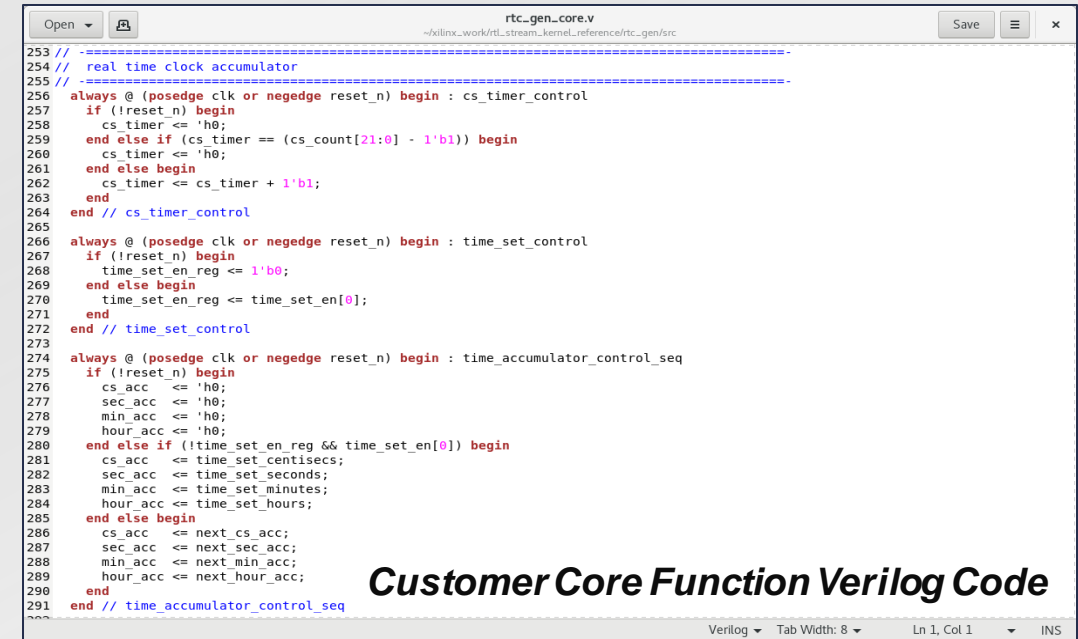
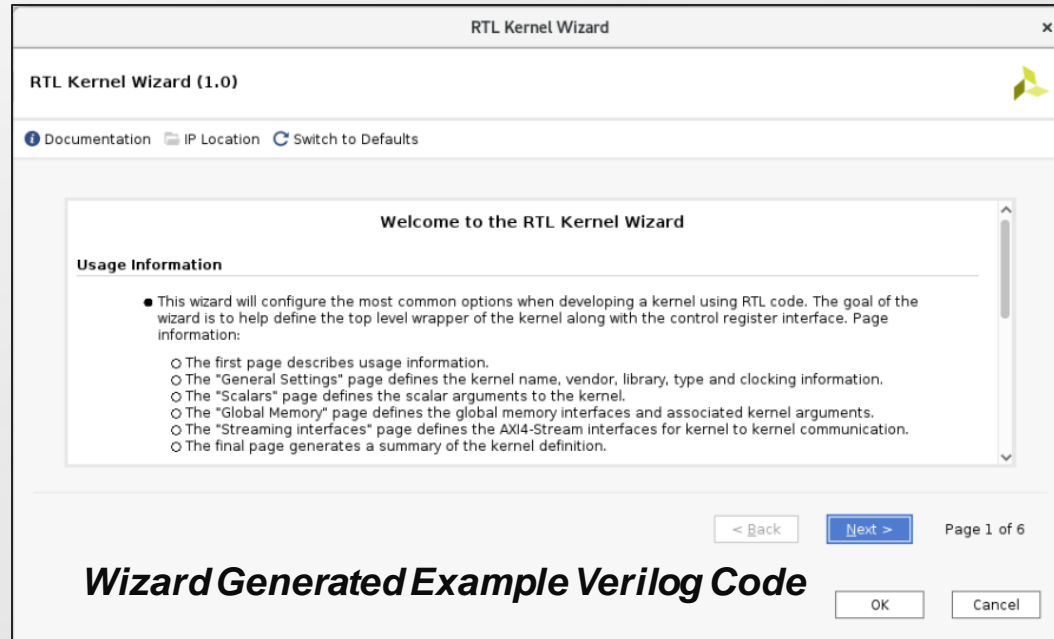
Vitis RTL Kernel

RTL Kernel

- *One AXI control slave port*
 - Control registers
 - Memory buffer pointers
 - Kernel start/stop control
- *Zero, one or more AXI master ports*
 - Read/write data buffer in on-board global memory
 - Read/write data buffer in host memory (with Slave Bridge support in the latest platforms)
- *Zero, one or more AXI stream ports*
 - Exchange data between kernels
 - Exchange data between kernel and host (with QDMA support in the latest platforms)
- *Core Function*



RTL Kernel Wizard



Vivado Kernel Packing Tool

Use **RTL Kernel Wizard** to start RTL kernel design easily!

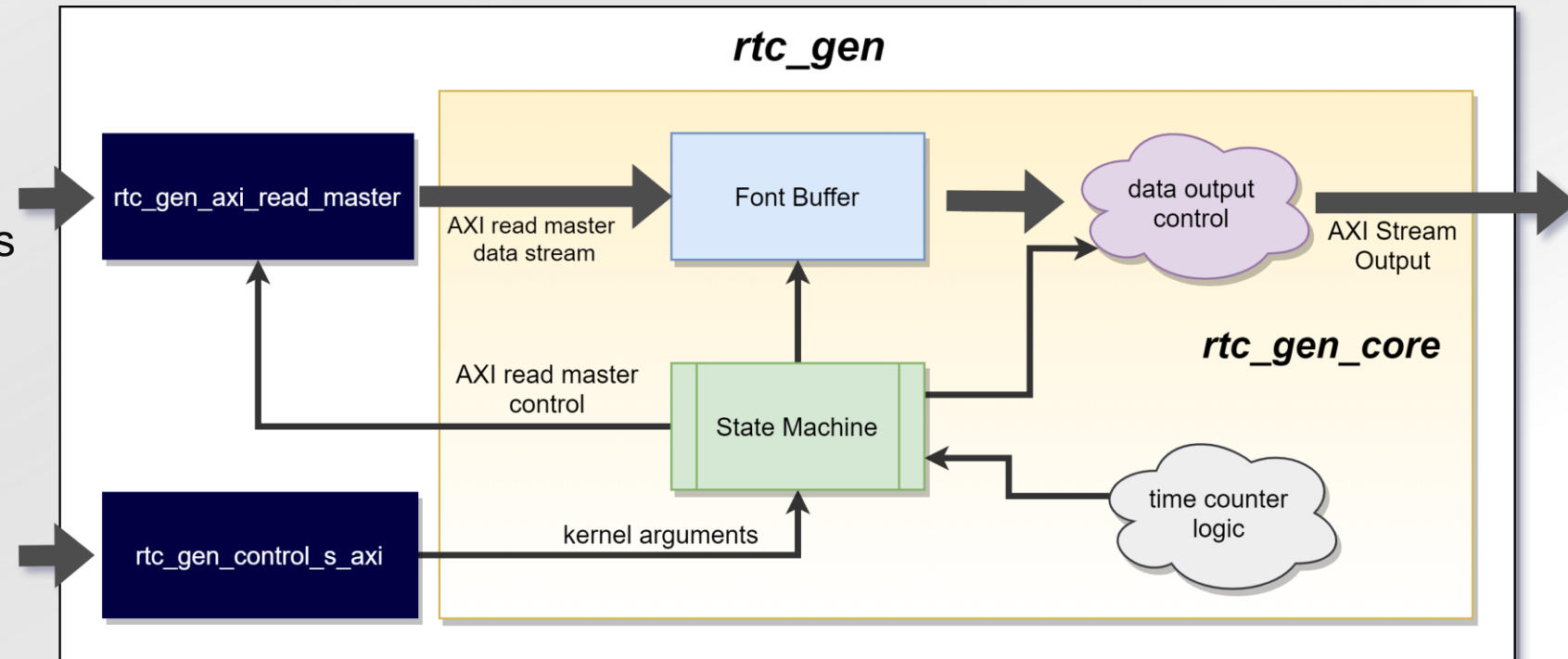
Vitis RTL Kernel

RTL Kernel Wizard will generate example AXI control slave, example AXI master, integration wrapper, example testbench, ...

Example RTL Kernel Design with AXI Stream

RTL Kernel *rtc_gen*

- One AXI control slave port
 - 6 kernel arguments
 - Control interface: *ap_ctrl_hs*
- One AXI master port
 - Data width: 32-bit
- One AXI stream master port
 - Data width: 64-bit



- Generated AXI master and AXI control slave modules are used.
- *rtc_gen_core* module is hand coded Verilog modules.
- Generated top level wrapper is modified to instantiate *rtc_gen_core* module.

File Edit View Search Terminal Help

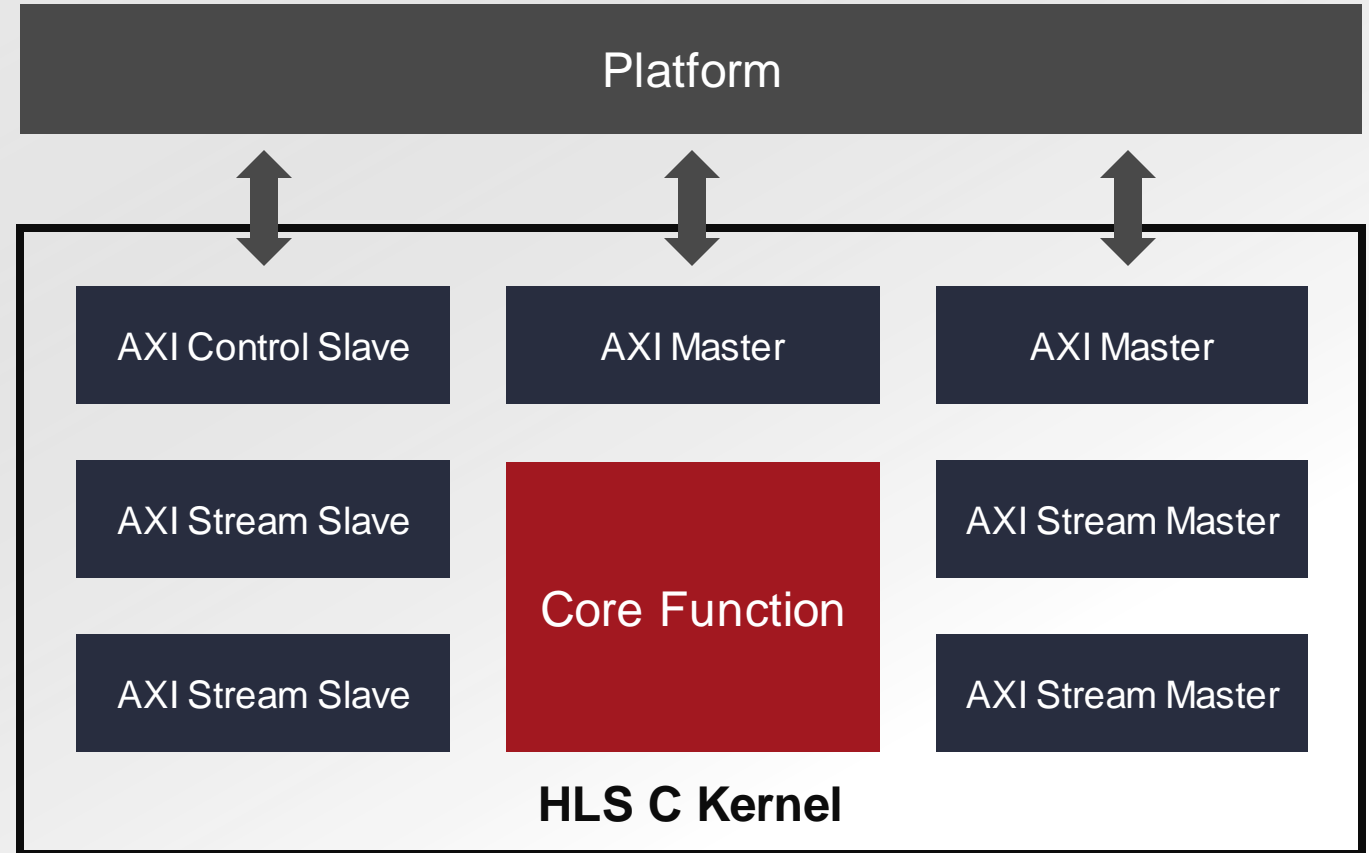
```
[gyuan@ws9800x vivado_project]$ vivado &  
[1] 30275  
[gyuan@ws9800x vivado_project]$ █
```

Create Vivado Project

HLS C Kernel

HLS C Kernel

- *One AXI control slave port*
 - *int* type function parameters
- *Zero, one or more AXI master ports*
 - *ap_uint** type function parameters
- *Zero, one or more AXI stream ports*
 - *hls::stream&* type function parameters
- *Core Function*



Vitis HLS and Accelerated Libraries

```
alpha_mix.cpp
103 }
104 }
105
106 // function to stream out the cv::Mat data
107 // input hls::stream data width is 24*8, output AXI stream data width is 64
108 void cvmat_stream_out(xf::cv::Mat<XF_BUC3, MAX_HEIGHT, MAX_WIDTH, XF_NPPC8> &input_img,
109                    hls::stream<ap_axiu<64, 0, 0, 0>> &stream_out)
110 {
111     for (int i = 0; i < input_img.rows * (input_img.cols >> XF_BITSHIFT(XF_NPP8)); i++)
112     {
113         ap_uint<24 * XF_NPPC8> data;
114         ap_axiu<64, 0, 0, 0> v;
115         data = input_img.read(i);
116         v.data = data.range(63,0);
117         stream_out.write(v);
118         v.data = data.range(127,64);
119         stream_out.write(v);
120         v.data = data.range(191,128);
121         stream_out.write(v);
122     }
123 }
124
125 // Top level kernel function
126 void alpha_mix(hls::stream<ap_axiu<64, 0, 0, 0>> &time_img_input, // time image input
127              ap_uint<32> *bgr_img_input, // background image input
128              hls::stream<ap_axiu<64, 0, 0, 0>> &mix_img_output, // mixed image output
129              int time_img_rows_in, // input time image height
130              int time_img_cols_in, // input time image width
131              int time_img_rows_rsz, // resized time image height
132              int time_img_cols_rsz, // resized time image width
133              int time_img_pos_row, // resized time image position - Y
134              int time_img_pos_col, // resized time image position - X
135              ap_uint<32> time_char_color, // [31:0] = [xRGB]
136              ap_uint<32> time_bgr_color, // [31:0] = [xRGB]
137              int time_bgr_opacity, // time image background opacity [7:0] is used
138              int bgr_img_rows, // background image height
139              int bgr_img_cols, // background image width
140              int bgr_img_cols)
141 }
```

Customer HLS C Code

Vitis Accelerated Libraries

Vitis™ Unified Software Platform includes an extensive set of open-source, performance-optimized libraries that offer out-of-the-box acceleration with minimal to zero-code changes to your existing applications.

Comprehensive documentation

- Common Vitis accelerated-libraries for Math, Statistics, Linear Algebra, and DSP offer a set of core functionality for a wide range of diverse applications.
- Domain-specific Vitis accelerated libraries offer out-of-the-box acceleration for workloads like Vision and Image Processing, Quantitative Finance, Database, and Data Analytics, Data Compression and more.
- Leverage the rich growing ecosystem of partner-accelerated libraries, framework plug-ins, and accelerated applications to hit the ground running and accelerate your path to production.

Vitis Accelerated Libraries

Use **Vitis HLS** to design and compile the HLS C code into Synthesizable Verilog code, and pack to Vitis Kernel

Vitis HLS

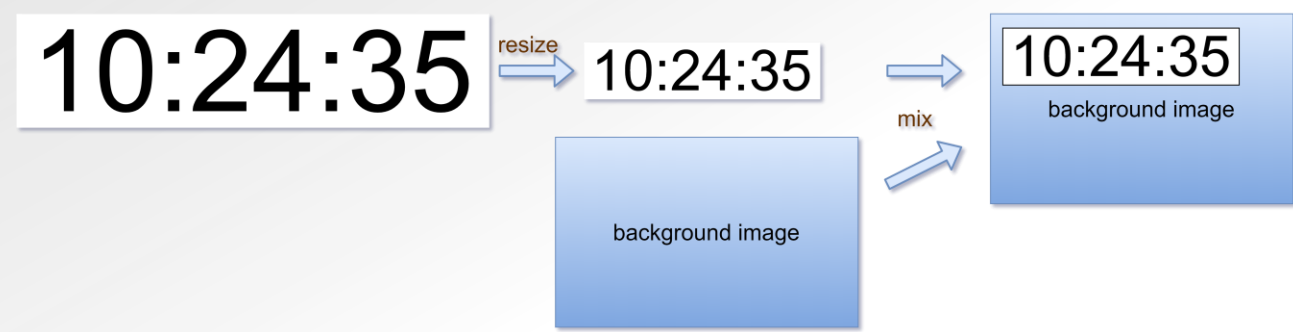
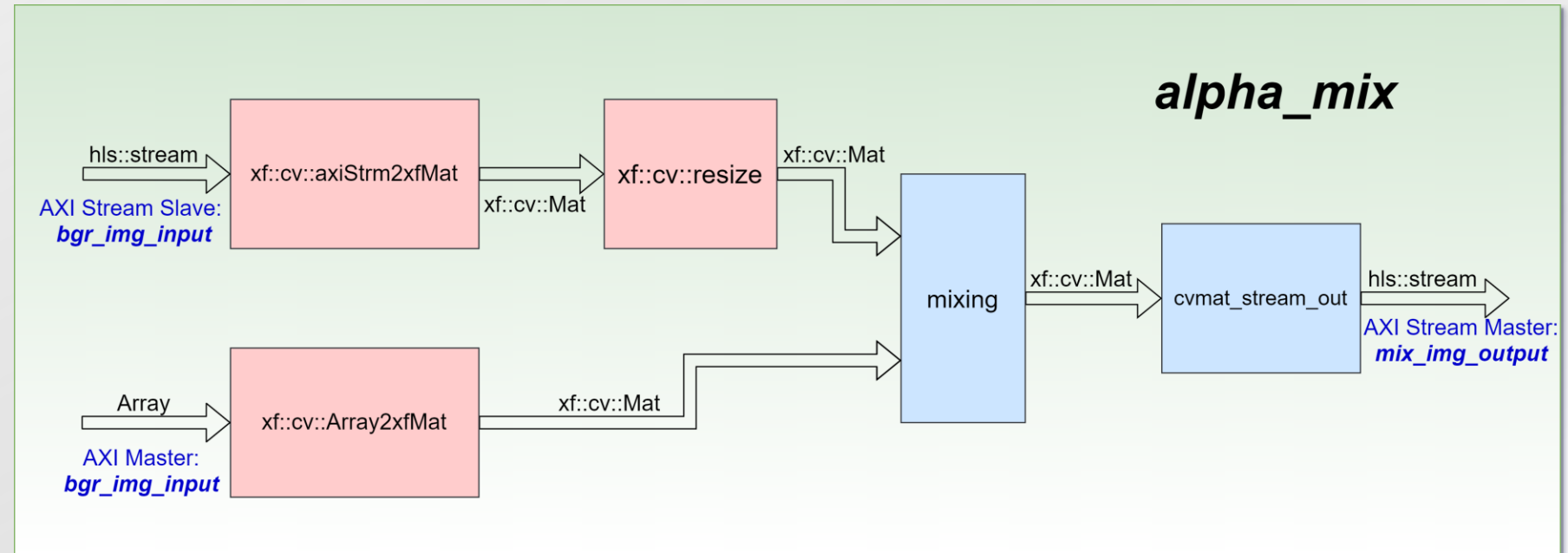
Vitis HLS Kernel

Vitis Accelerated Libraries provide a rich set of fully verified designs, which can easily used to reduce the design cycle.

Example HLS C Kernel Design with Vitis Vision Library

HLS C kernel: *alpha_mix*

- Top function
 - *alpha_mix*
- AXI Control Slave
 - *alpha_mix*
- AXI Master
 - *bgr_img_input*
- AXI Stream
 - Slave: *time_img_input*
 - Master: *mix_img_out*
- Vitis Vision Library Functions
 - `xf::cv::axiStrm2xfMat`
 - `xf::cv::Array2xfMat`
 - `xf::cv::resize`
- User Functions
 - *mixing*
 - *cvmat_stream_out*



Example HLS C Kernel Design with Vitis Vision Library

```
void alpha_mix(hls::stream<ap_axiu<64, 0, 0, 0>> &time_img_input, // time image input
              ap_uint<512> *bgr_img_input, // background image input
              hls::stream<ap_axiu<64, 0, 0, 0>> &mix_img_output, // mixed image output
              int time_img_rows_in, // input time image height
              int time_img_cols_in, // input time image width
              int time_img_rows_rsz, // resized time image height
              int time_img_cols_rsz, // resized time image width
              int time_img_pos_row, // resized time image position - Y
              int time_img_pos_col, // resized time image position - X
              ap_uint<32> time_char_color, // [31:0] = [xRGB]
              ap_uint<32> time_bgr_color, // [31:0] = [xRGB]
              int time_bgr_opacity, // time image background opacity. [7:0] used
              int bgr_img_rows, // background image height
              int bgr_img_cols // background image width
              )
```

Top level kernel function definition of alpha_mix.

```
v++ --platform xilinx_u50_gen3x16_xdma_201920_3 \
    --target hw \
    --kernel alpha_mix \
    --include ./include \
    --advanced.prop kernel.alpha_mix.kernel_flags="-std=c++0x -D__SDSVHLS__ -DHLS_NO_XIL_FPO_LIB" \
    --compile \
    --output alpha_mix.xo \
    alpha_mix.c
```

Vitis compiling command line.



Example Simple HLS C Kernel Design

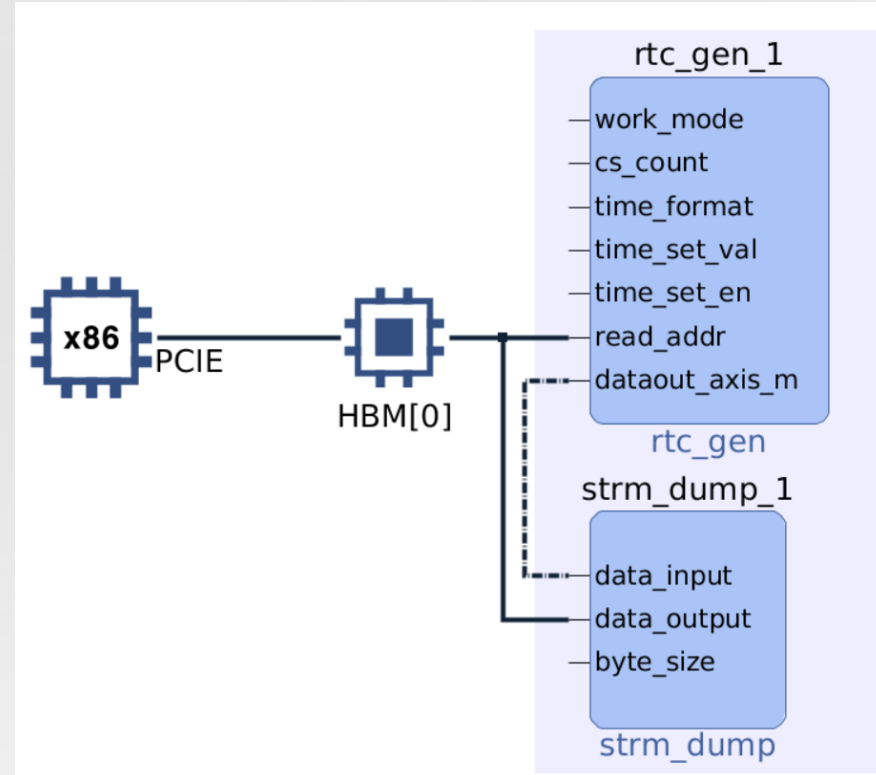
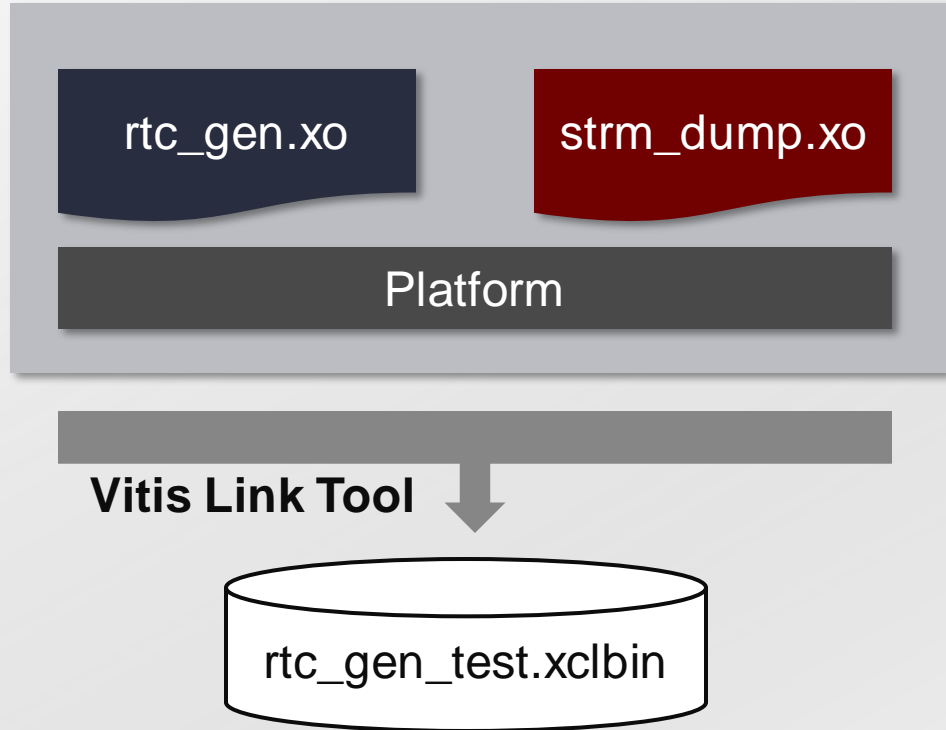
```
extern "C"
{
    // kernel function.
    void strm_dump (hls::stream<ap_axiu<INPUT_PTR_WIDTH, 0, 0, 0>>& data_input,
                   ap_uint<OUTPUT_PTR_WIDTH> *data_output,
                   int byte_size)
    {
        for (int i = 0; i < (byte_size / 8); i++) {
            // clang-format off
            #pragma HLS PIPELINE II = 1
            // clang-format on
            ap_axiu<INPUT_PTR_WIDTH, 0, 0, 0> temp = data_input.read();
            data_output[i] = temp.data;
        }
    }
}
```

Simple HLS kernel: *strm_dump*

- *Simple module to convert AXI stream slave port to AXI Lite Master port*



Use Vitis to Build Hardware Overlay (XCLBIN)

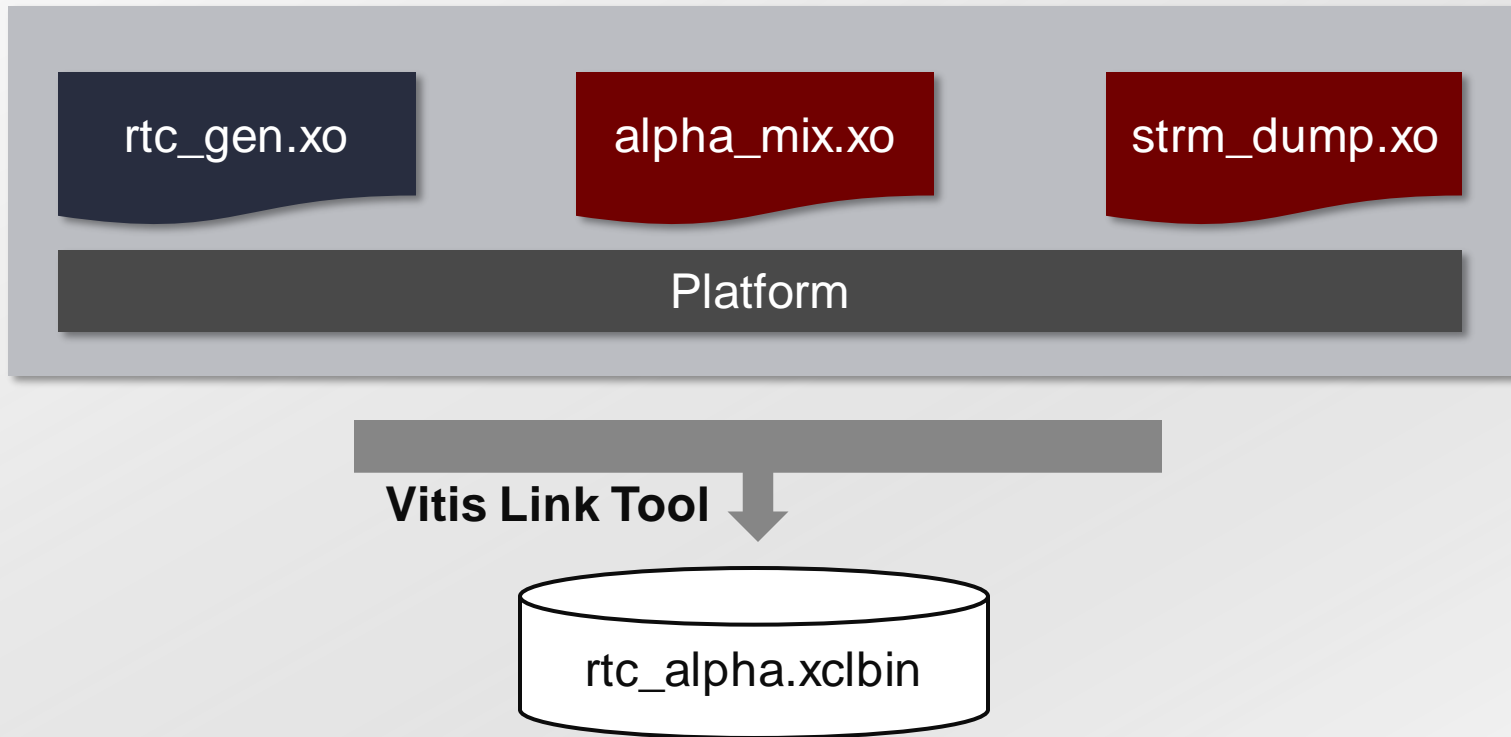


```
v++ -l \
  -platform xilinx_u50_gen3x16_xdma_201920_3 \
  -config xclbin_rtc_gen_test.ini \
  -o rtc_gen_test.xclbin \
  rtc_gen.xo strm_dump.xo
```

xclbin_rtc_gen_test.ini

```
[connectivity]
stream_connect=rtc_gen_1.dataout_axis_m:strm_dump_1.data_input
```

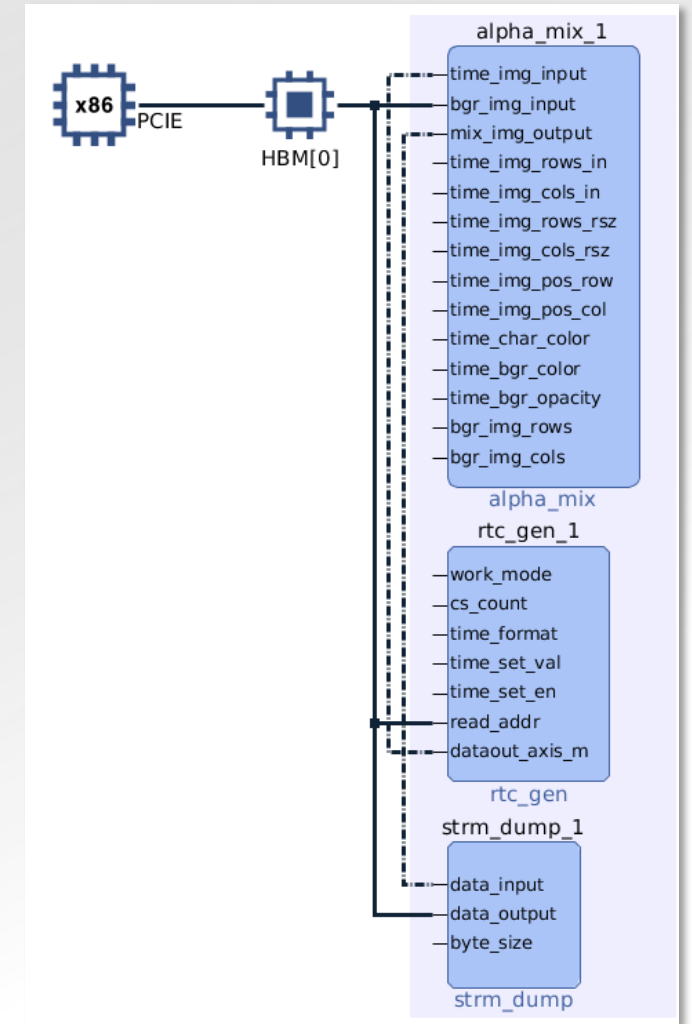
Use Vitis to Build Hardware Overlay (XCLBIN)



```
v++ -l \
  -platform xilinx_u50_gen3x16_xdma_201920_3 \
  -config xclbin_rtc_alpha.ini \
  -o rtc_alpha.xclbin \
  rtc_gen.xo strm_dump.xo alpha_mix.so
```

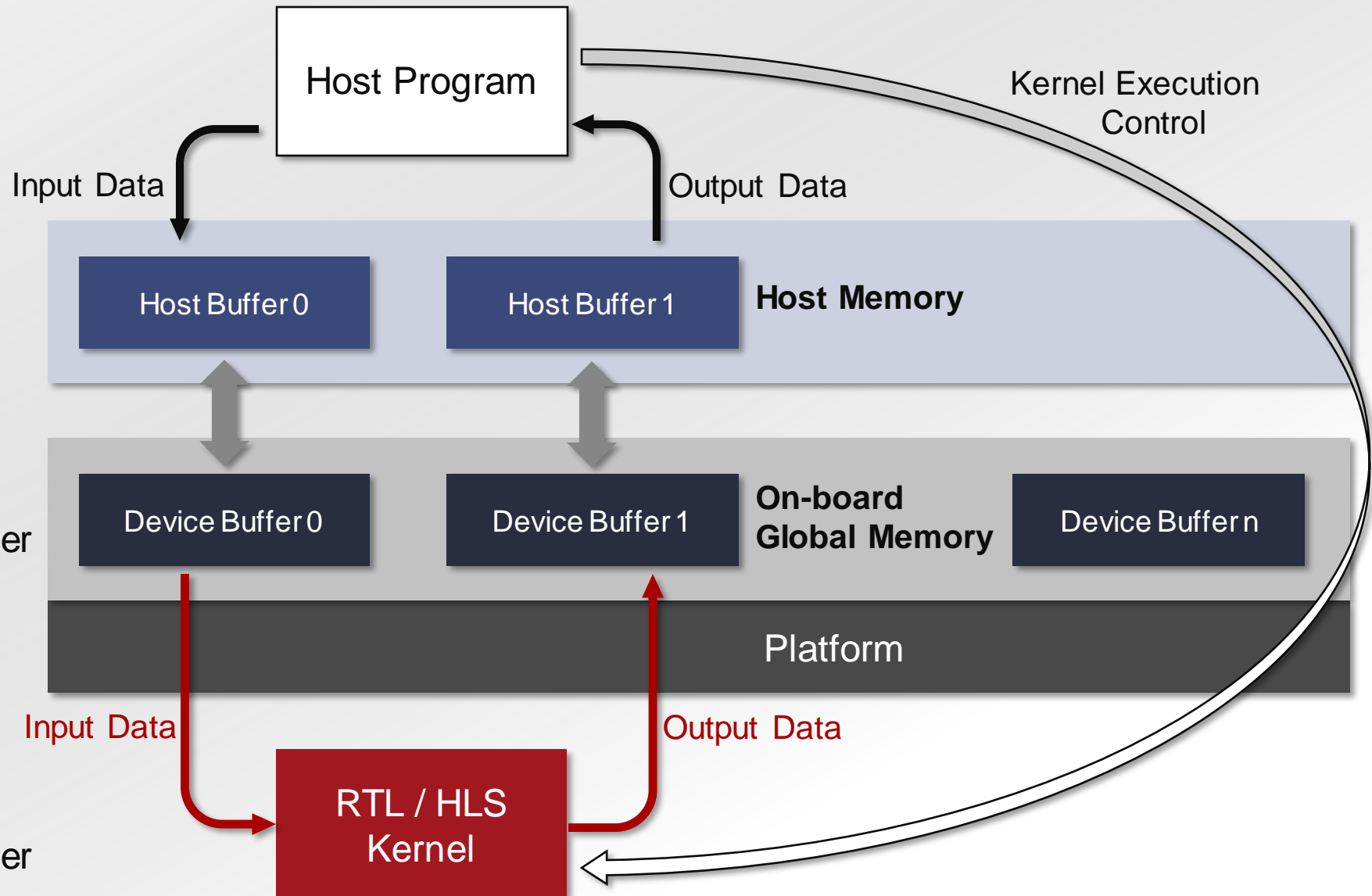
xclbin_rtc_alpha.ini

```
[connectivity]
stream_connect=rtc_gen_1.dataout_axis_m:alpha_mix_1.time_img_input
stream_connect=alpha_mix_1.mix_img_output:strm_dump_1.data_input
```



Write Host-side Programs

1. Program FPGA
2. Identify Kernel
3. Allocate host buffer
4. Allocate device buffer
5. Set Kernel arguments
6. Migrate input data from host buffer to device buffer
7. Trigger Kernel Execution
8. Waiting for Kernel Finish
9. Migrate output data from device buffer to host buffer

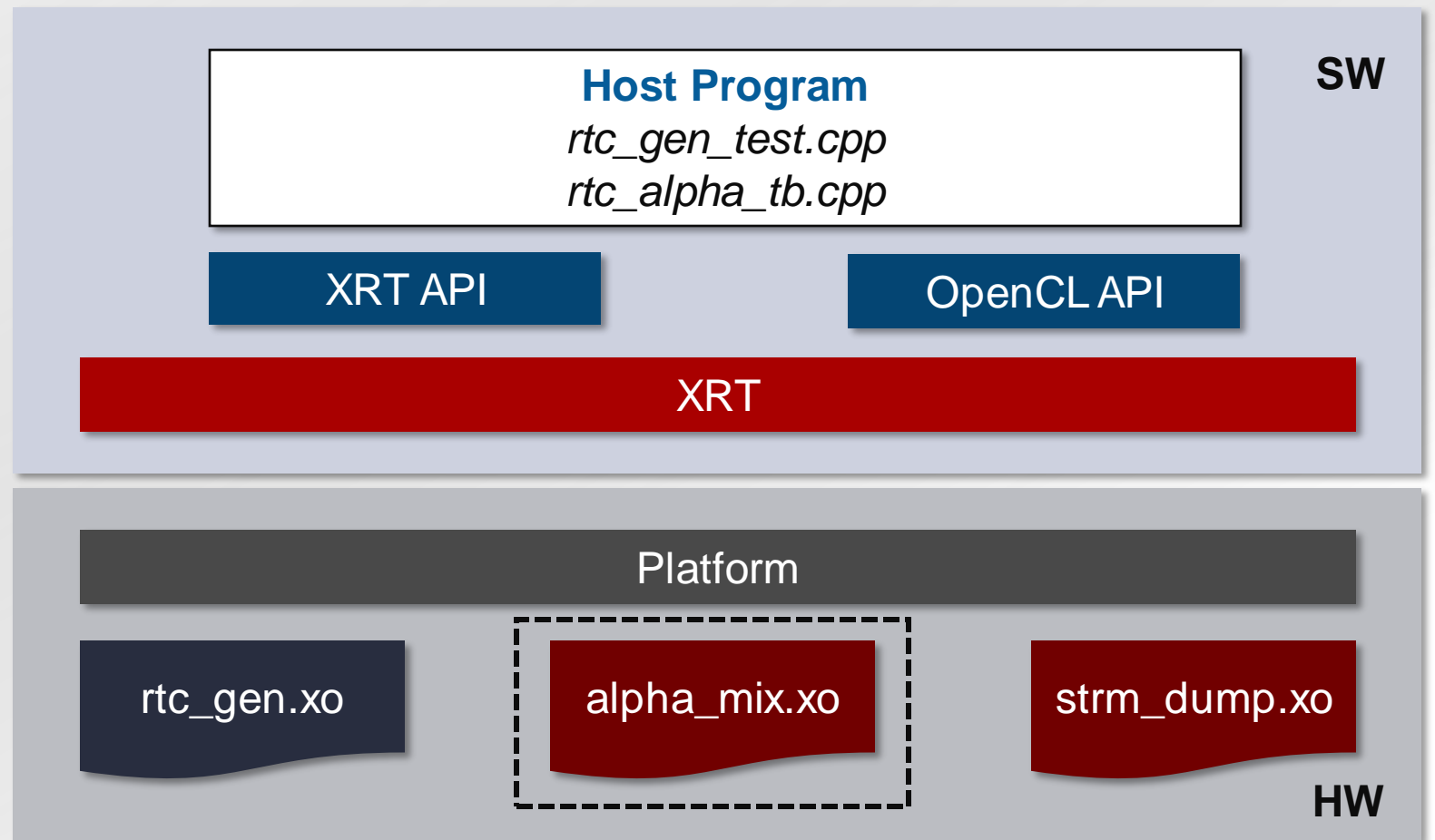


Example Host Program with Different Level of API

Host program:

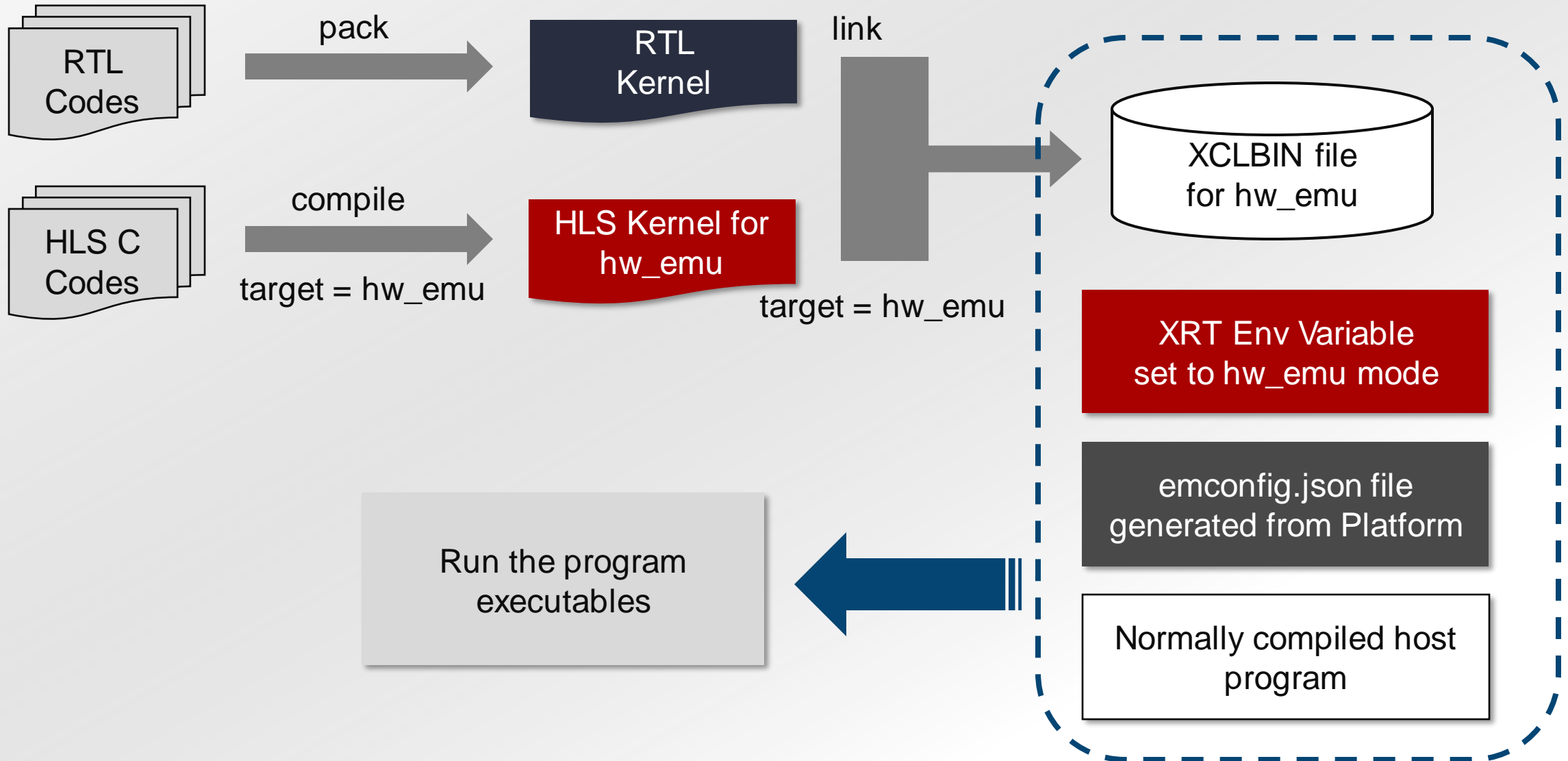
rtc_gen_test.cpp
rtc_alpha_tb.cpp

- Use low level API for XCLBIN file loading and direct control register access
- Use high level OpenCL API for kernel arguments setting and execution control
- Support both hardware target and hardware emulation target



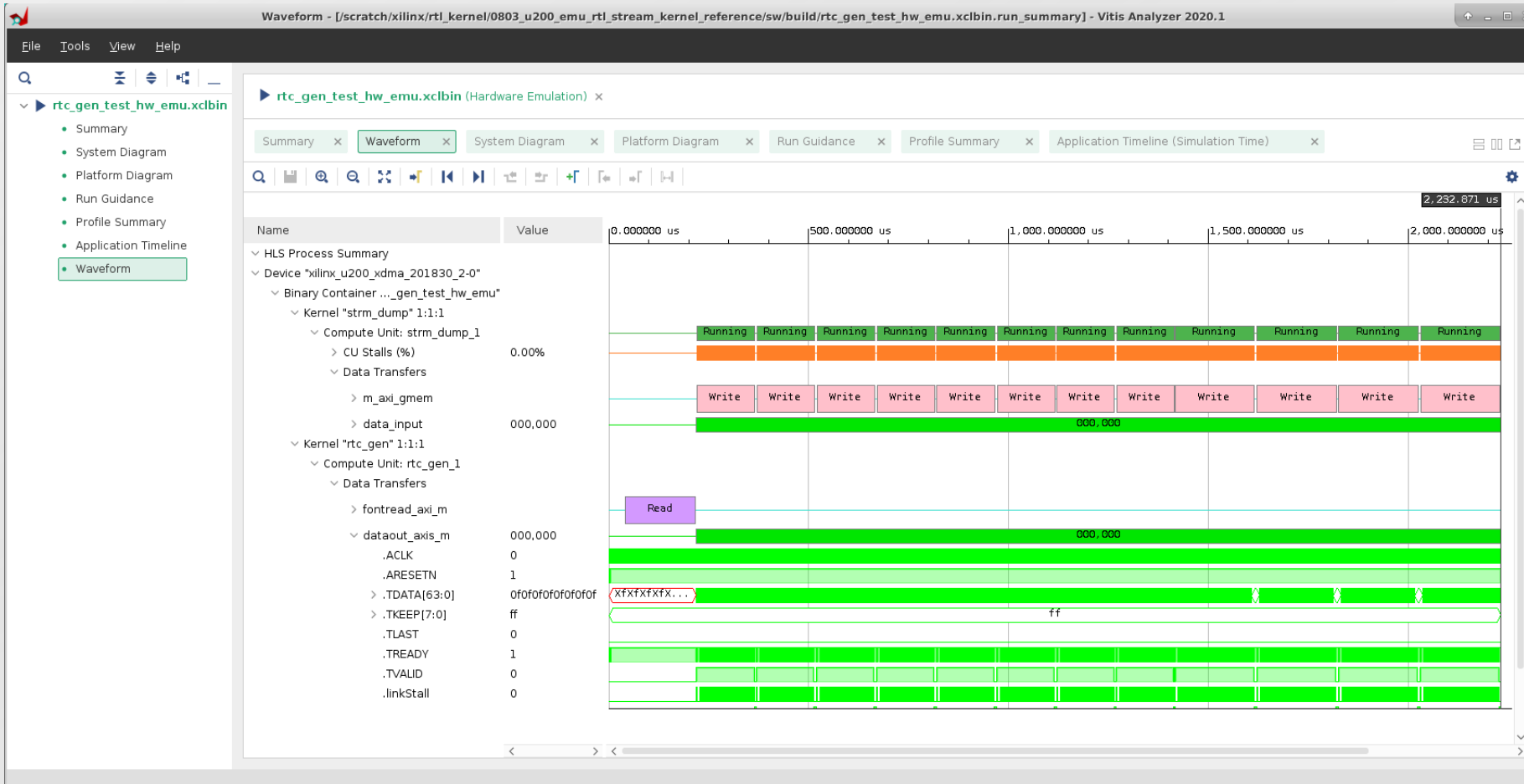
In the example host program, the usage of low-level API *xcIRegWrite* / *xcIRegRead* is illustrated. User can use these two functions to access the AXI slave register directly, which might be valuable for specific control schemes of some traditional RTL design.

Use Emulation to Verify Mixed Kernel Designs



Use Emulation to Verify Mixed Kernel Designs

Hardware Emulation – provides insight to the interaction details of the kernels, platform and software



Use Vitis Analyzer to Profile the System Designs

Profiling – provides inclusive statistics information for software and kernel execution

The screenshot shows the Vitis Analyzer 2020.1 interface. The title bar indicates the profile summary for a system run. The sidebar on the left lists various views, with 'Profile Summary' selected. The main content area is divided into several sections:

- Kernels & Compute Units**: A summary table showing kernel execution statistics.
- Kernel Execution**: A table with columns for Kernel, Enqueues, Total Time (ms), Min Time (ms), Avg Time (ms), and Max Time (ms).
- Top Kernel Execution**: A table with columns for Kernel, Kernel Instance Address, Context ID, Command Queue ID, Device, Start Time (ms), and Duration (ms).
- Compute Unit Utilization**: A table with columns for Compute Unit, Kernel, Device, Calls, Dataflow Execution, Max Parallel Executions, Dataflow Acceleration, CU Utilization (%), Total Time (ms), Min Time (ms), and Avg Time (ms).

Kernel	Enqueues	Total Time (ms)	Min Time (ms)	Avg Time (ms)	Max Time (ms)
alpha_mix	74	110.099	1.468	1.488	1.927
rtc_gen	75	64.760	0.579	0.863	1.958
strm_dump	74	106.949	1.401	1.445	1.903

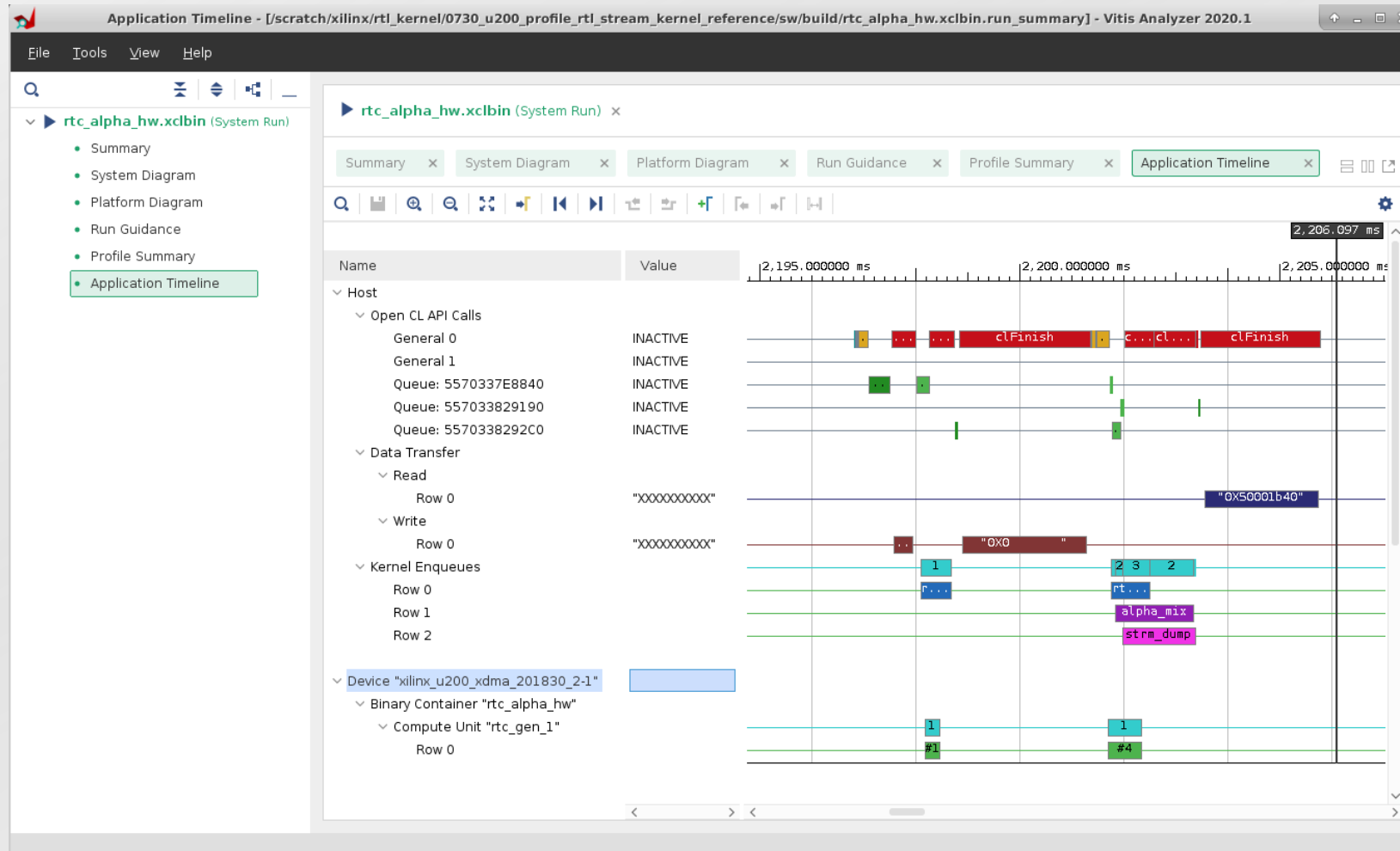
Kernel	Kernel Instance Address	Context ID	Command Queue ID	Device	Start Time (ms)	Duration (ms)
rtc_gen	0x557033987570	0	0	xilinx_u200_xdma_201830_2-1	7035.980	1.958
alpha_mix	0x5570338f51c0	0	2	xilinx_u200_xdma_201830_2-1	7036.140	1.927
strm_dump	0x557033855760	0	1	xilinx_u200_xdma_201830_2-1	7036.240	1.903
alpha_mix	0x5570338f51c0	0	2	xilinx_u200_xdma_201830_2-1	9500.640	1.541
alpha_mix	0x5570338f51c0	0	2	xilinx_u200_xdma_201830_2-1	6287.630	1.538
alpha_mix	0x5570338f51c0	0	2	xilinx_u200_xdma_201830_2-1	8538.770	1.532
alpha_mix	0x5570338f51c0	0	2	xilinx_u200_xdma_201830_2-1	2966.000	1.531
alpha_mix	0x5570338f51c0	0	2	xilinx_u200_xdma_201830_2-1	7143.940	1.529
alpha_mix	0x5570338f51c0	0	2	xilinx_u200_xdma_201830_2-1	7679.740	1.528
alpha_mix	0x5570338f51c0	0	2	xilinx_u200_xdma_201830_2-1	4679.950	1.525

Compute Unit	Kernel	Device	Calls	Dataflow Execution	Max Parallel Executions	Dataflow Acceleration	CU Utilization (%)	Total Time (ms)	Min Time (ms)	Avg Time (ms)
alpha_mix_1	alpha_mix	xilinx_u200_xdma_201830_2-1	74	Yes	1	1.000000x	1.313	102.936	1.338	
rtc_gen_1	rtc_gen	xilinx_u200_xdma_201830_2-1	75	No	1	1.000000x	0.588	46.100	0.265	
strm_dump_1	strm_dump	xilinx_u200_xdma_201830_2-1	74	Yes	1	1.000000x	1.23	96.449	1.296	



Use Vitis Analyzer to Profile the System Designs

Profiling – provides inclusive statistics information for software and kernel execution





Home



Trash

```

Terminal
File Edit View Search Terminal Help
[MESSAGE] Reading font data finished
[MESSAGE] Background Image ../media/alveo.jpg Read in: 1024 x 512
[MESSAGE] Font data loaded to Alveo global memory
[MESSAGE] Font data loaded to FPGA on-chip SRAM
[MESSAGE] Background image loaded to Alveo global memory
[MESSAGE] Program exit normally.
[gyuan@ws9800x build]$ ./rtc_gen_test
[MESSAGE] Program running in hardware mode
[MESSAGE] XCLBIN
01830_2
Found Platform
Platform Name: Xil
[MESSAGE] found pl
[MESSAGE] target c
INFO: Reading rtc
Loading: 'rtc_gen
[MESSAGE] CU inde
[MESSAGE] FPGA ini
-----
[MESSAGE] Font lit
[MESSAGE] Reading
[MESSAGE] Font dat
[MESSAGE] Font dat

```

Real Time Clock - 8 digits

23:27:43



Thank you!

