XILINX

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

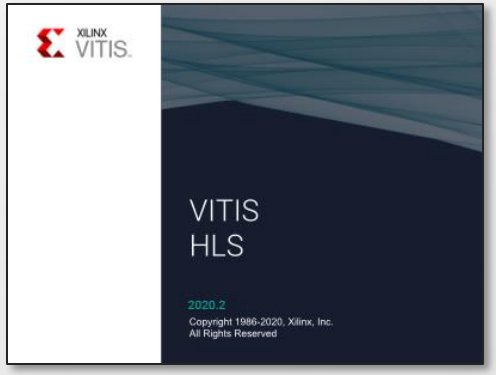
Alveo U200

Alveo U250

Alveo U280

XILINX

Inclusive Hardware Design Methodology



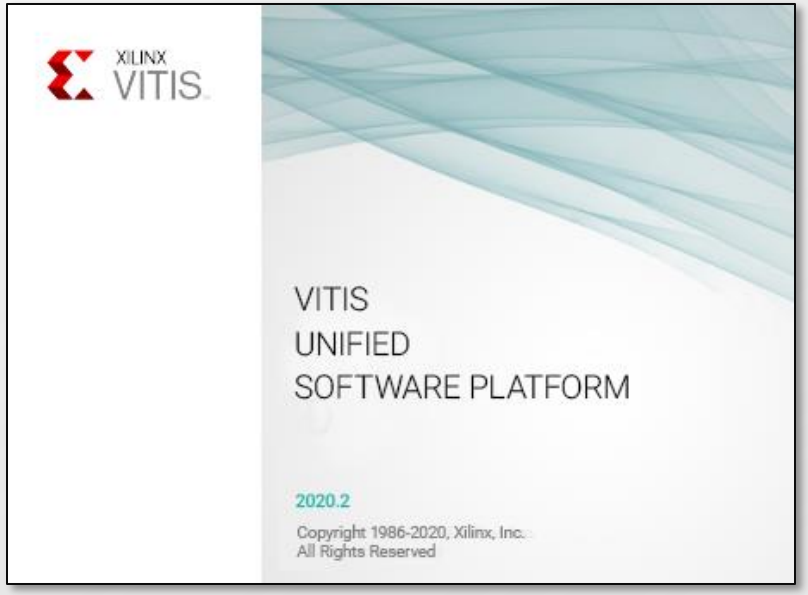
High Level Synthesis (HLS) Modules

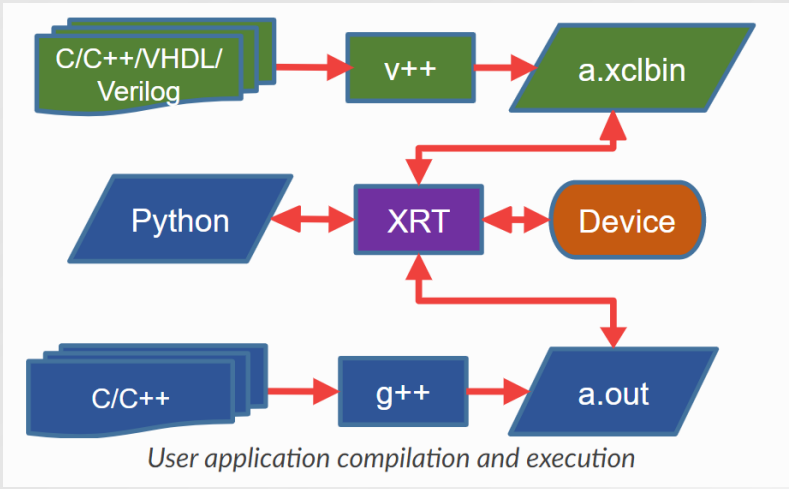
- Image Processing
- Matrix Operation
- Encryption
- Data Compression
- Others..

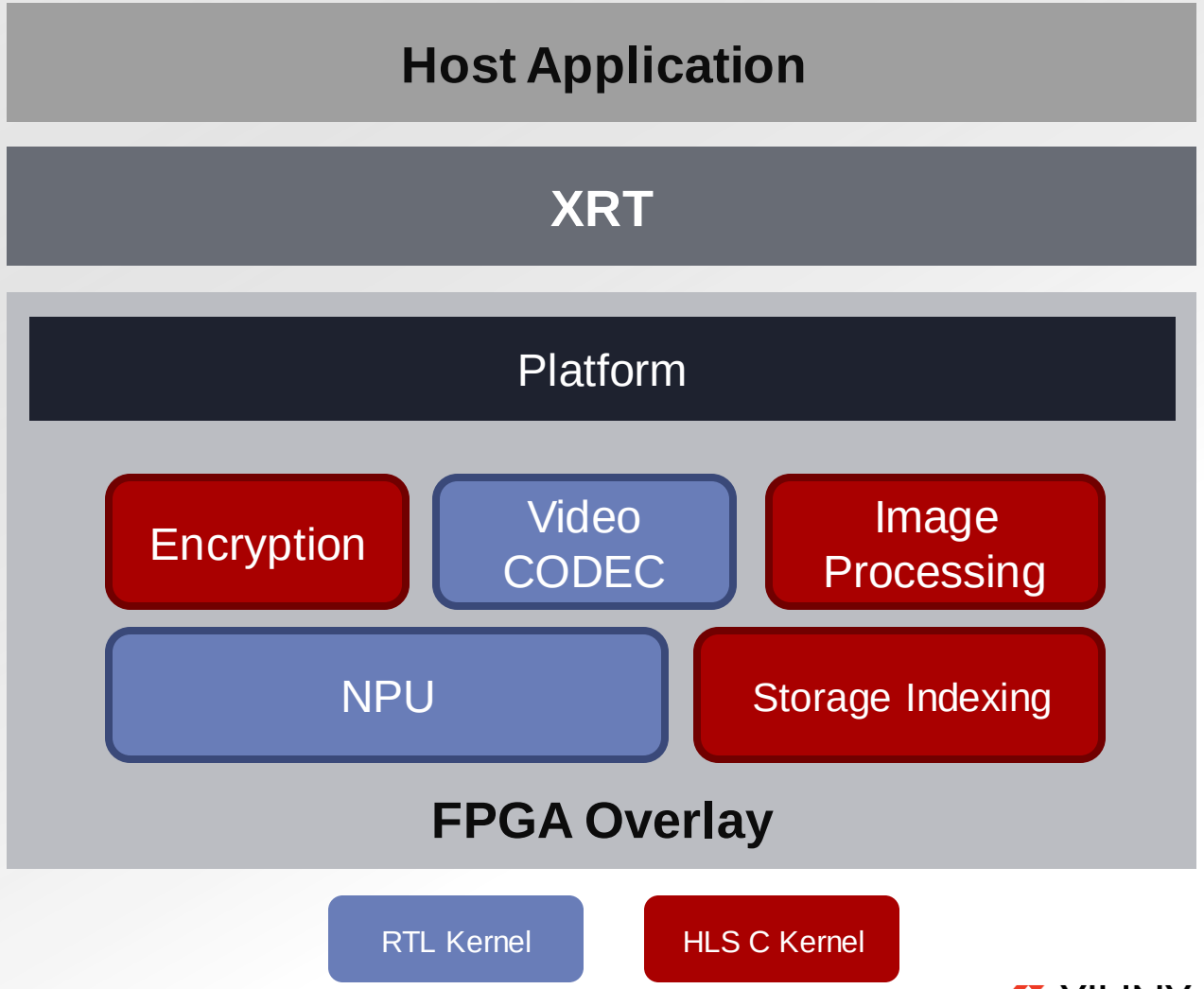
RTL Modules

- Video CODEC
- Image CODEC
- Processor
- NPU
- Others..

Fast Mixed Kernel Integration Flow with Vitis





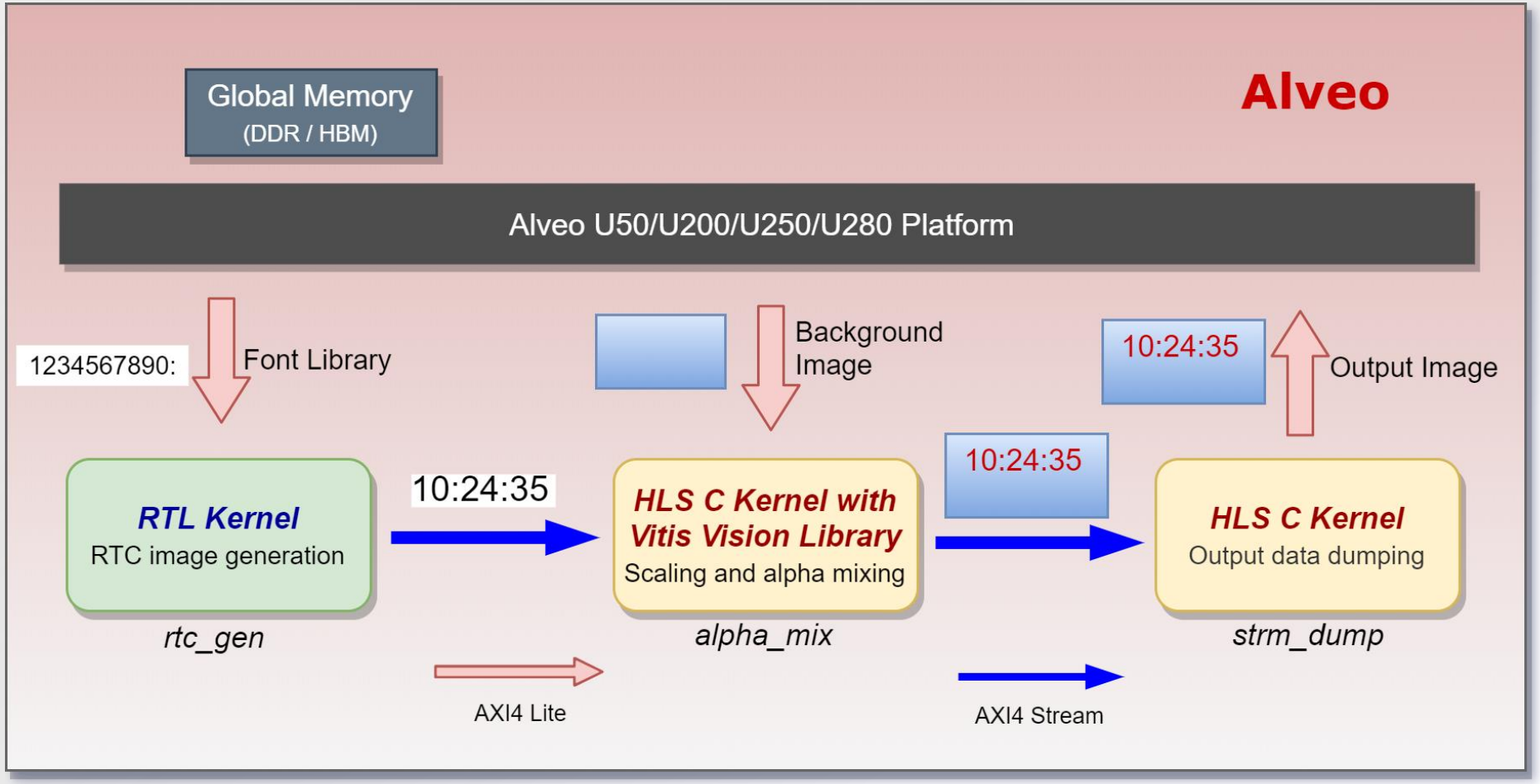


XILINX

Mixed Kernel Design Example in Vitis Tutorials

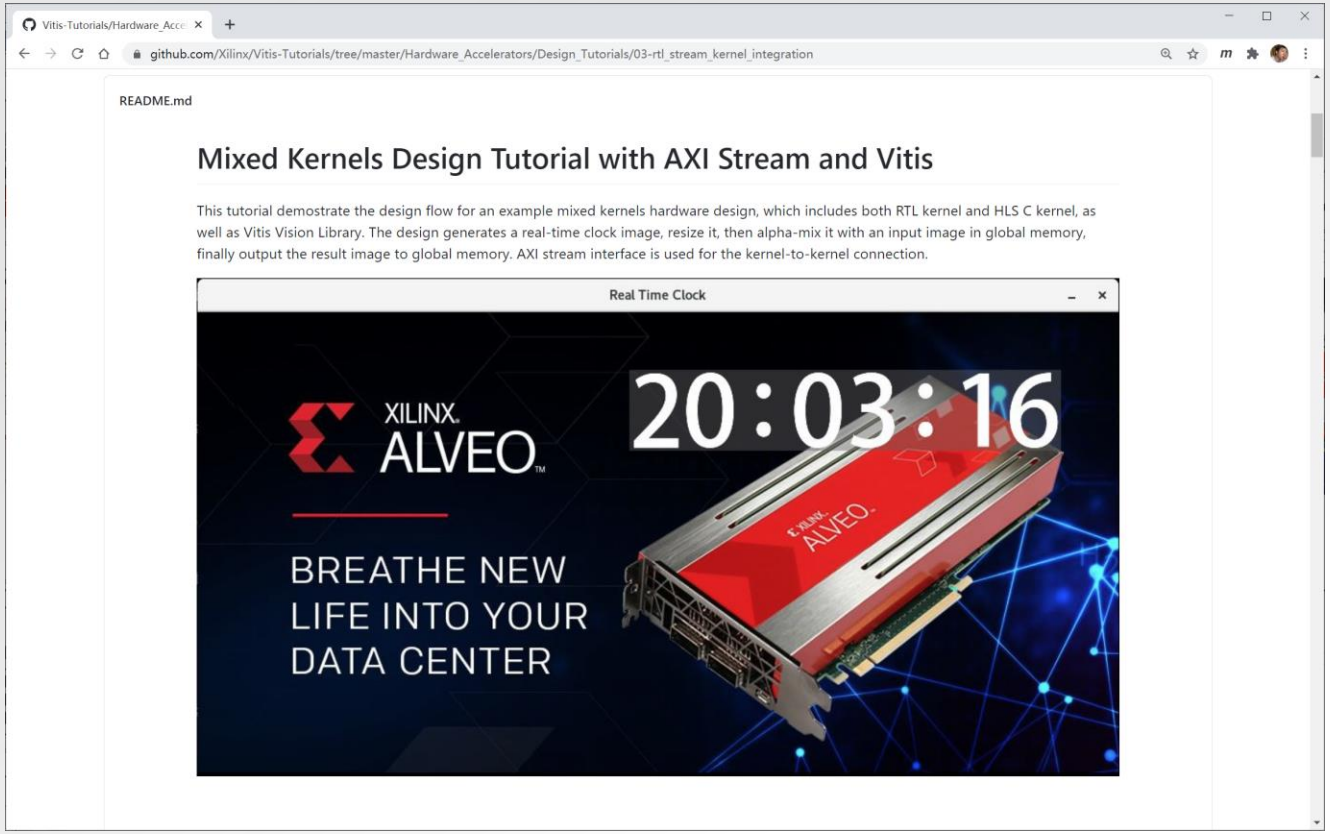
Mixed HDL/HLS Kernel Integration Example

- ***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



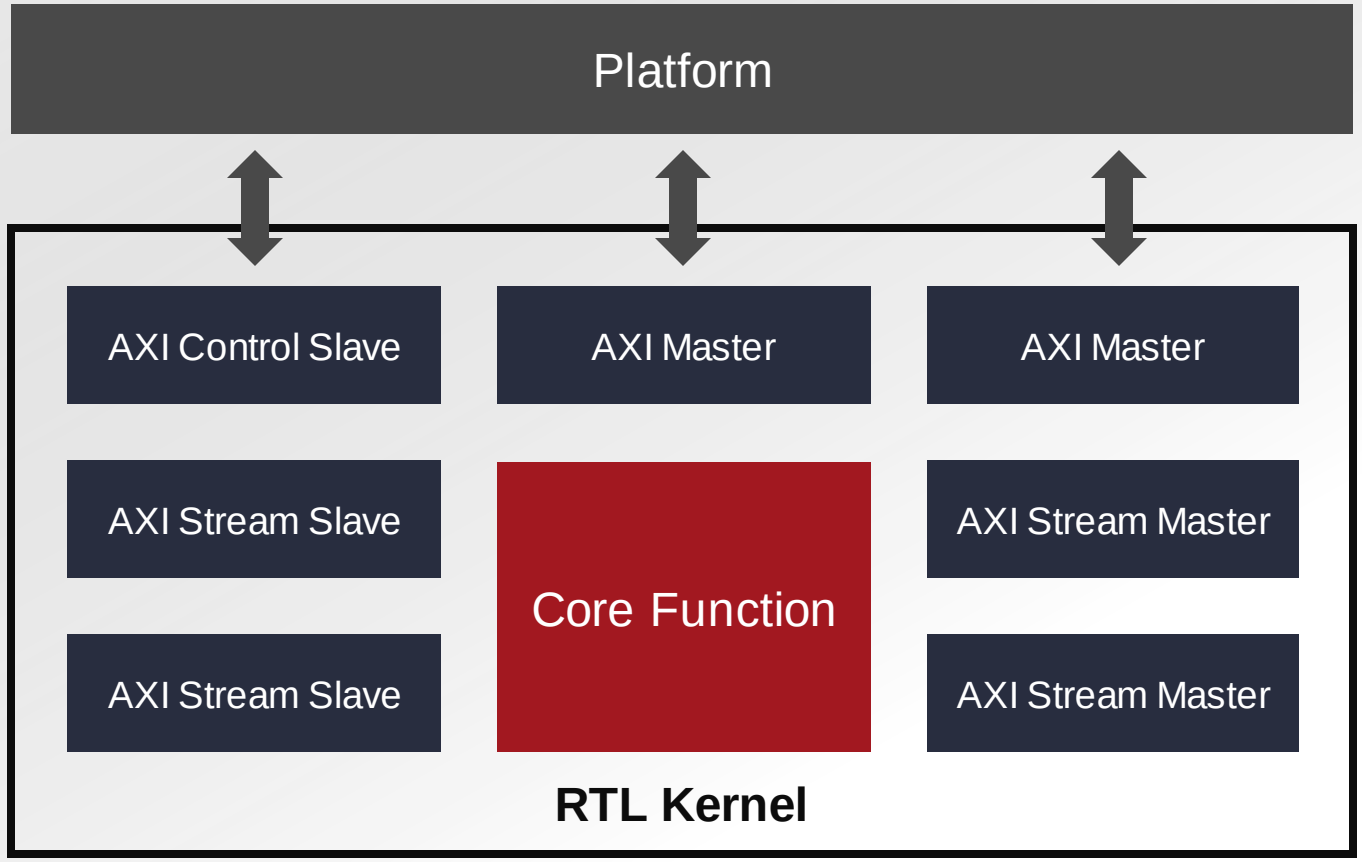
Linux git command lines to get the example design

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

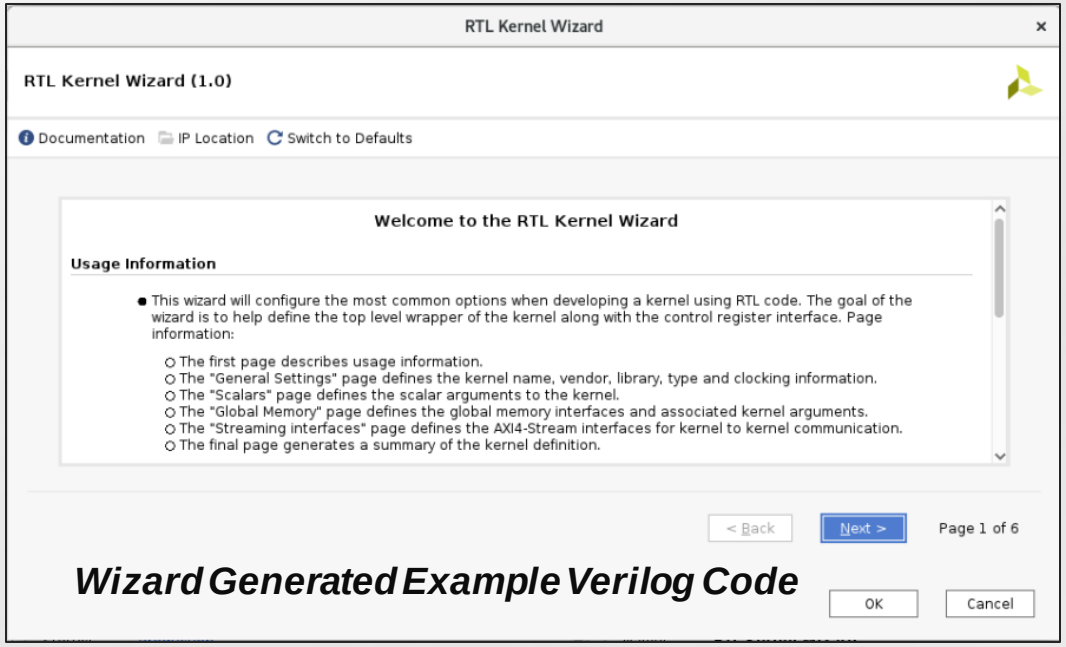
Vitis RTL Kernel

RTL Kernel

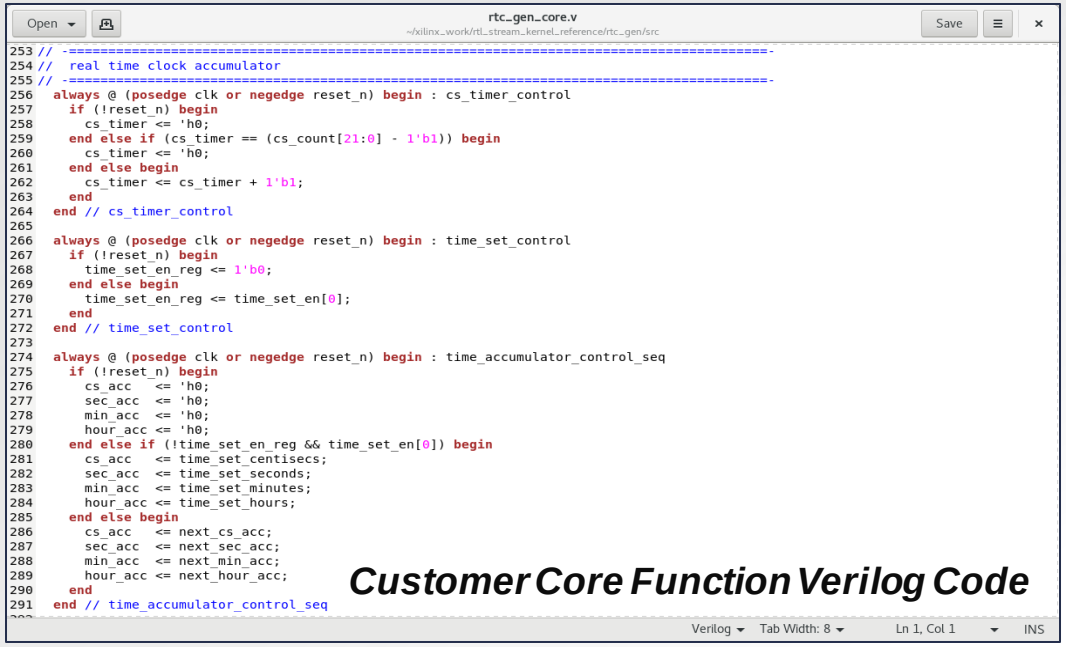
- *One AXI control slave port*
  - Control registers
  - Memory buffer pointers
  - Kernel start/stop control
- *Zero, one or more AXI master ports*
  - Read/write data buffer in on-board global memory
  - Read/write data buffer in host memory (with Slave Bridge support in the latest platforms)
- *Zero,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



Wizard Generated Example Verilog Code



Customer Core Function Verilog Code

Vivado Kernel Packing Tool

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

Vitis RTL Kernel

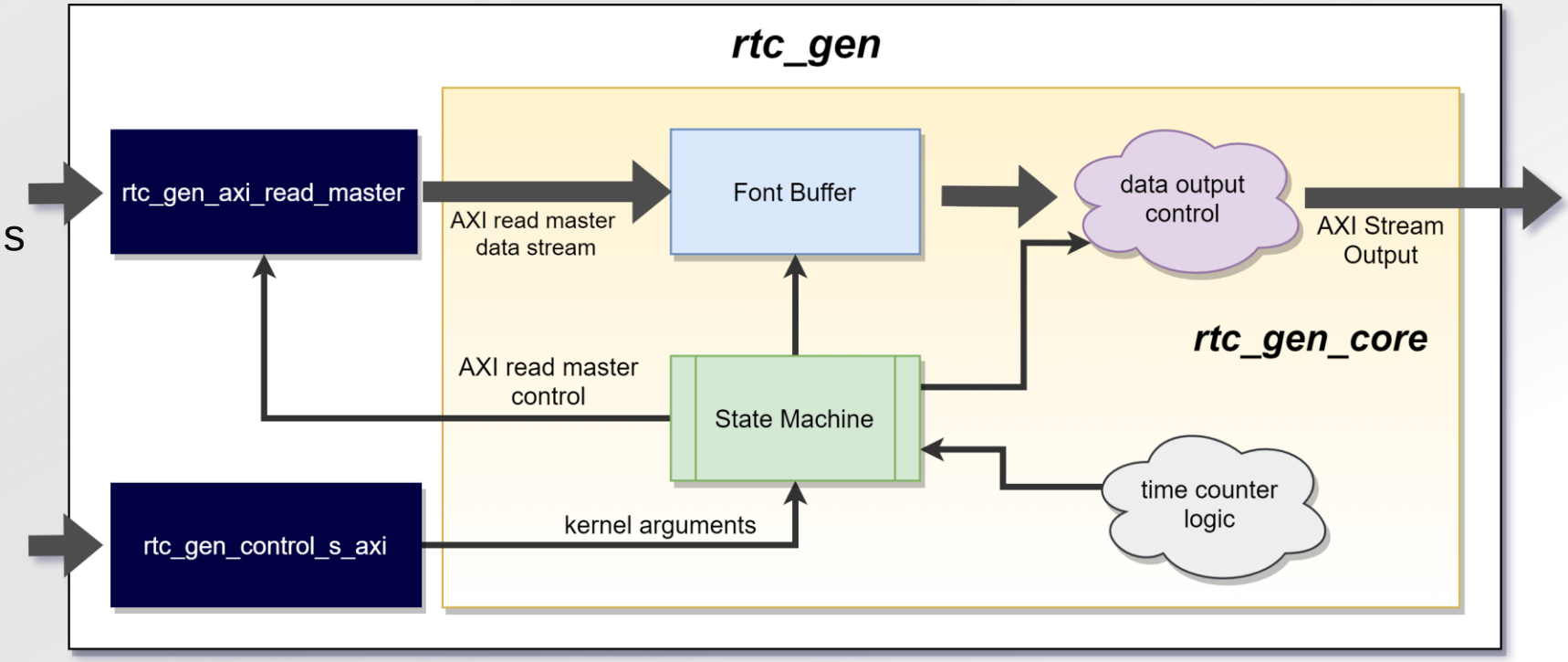
RTL Kernel Wizard will generate example AXI control slave, example AXI master, integration wrapper, example testbench, …

XILINX

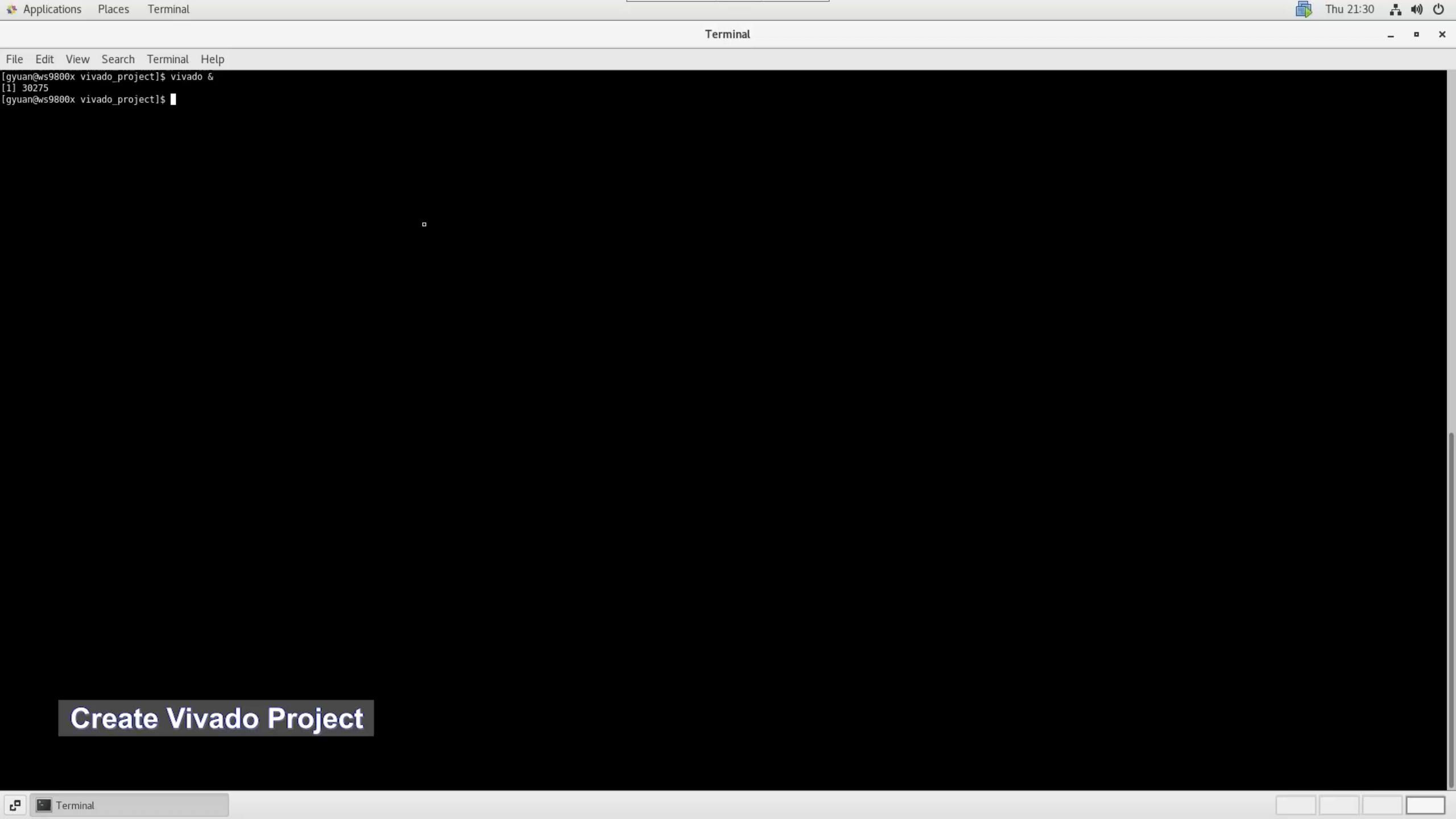
Example RTL Kernel Design with AXI Stream

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



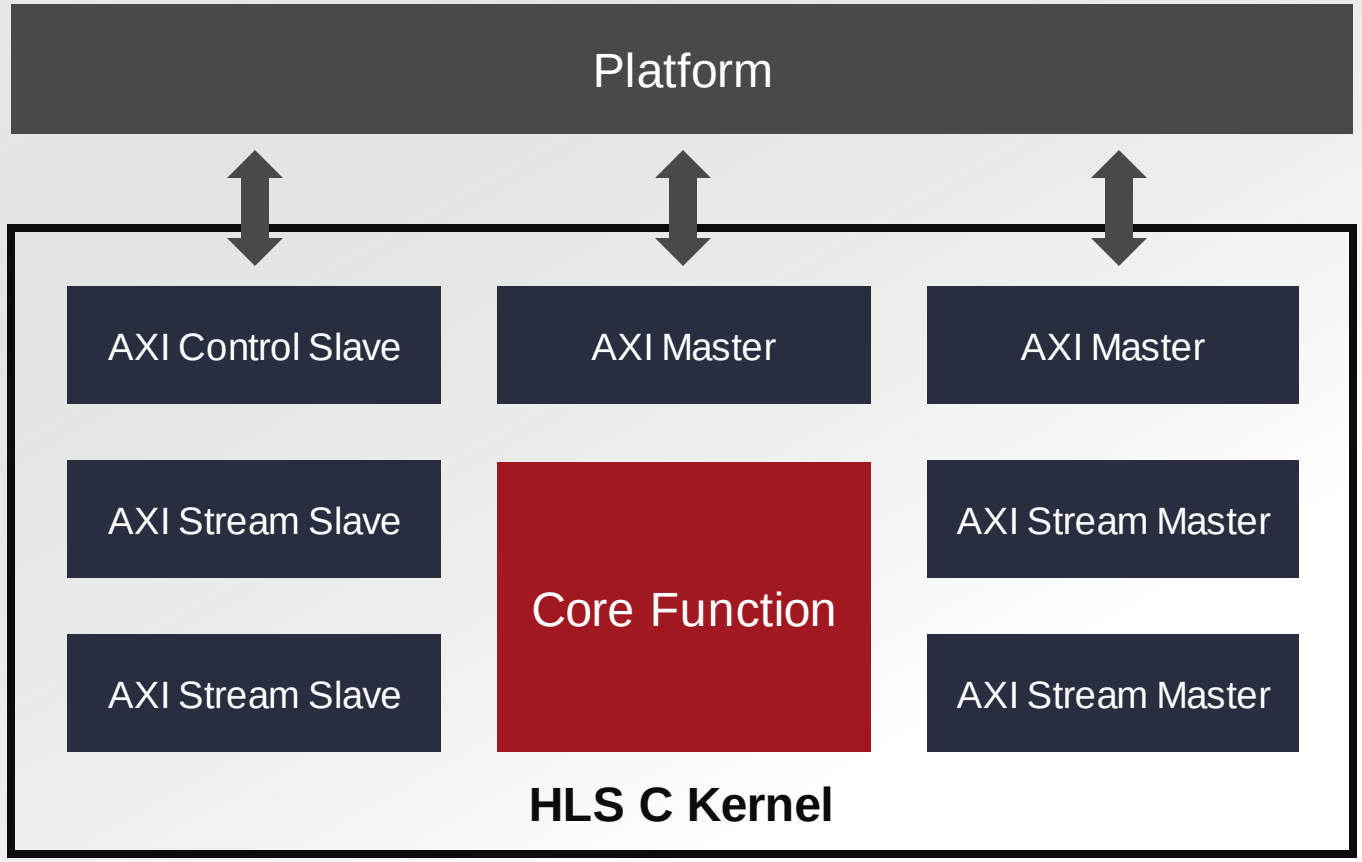
- Generated AXI master and AXI control slave modules are used.
- ***rtc_gen_core*** module is hand coded Verilog modules.
- Generated top level wrapper is modified to instantiate ***rtc_gen_core*** module.


Applications
Places
Terminal
Thu 21:30
Terminal
File
Edit
View
Search
Terminal
Help
[gyuan@ws9800x vivado_project]$ vivado &
[1] 30275
[gyuan@ws9800x vivado_project]$
Create Vivado Project
Terminal

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*

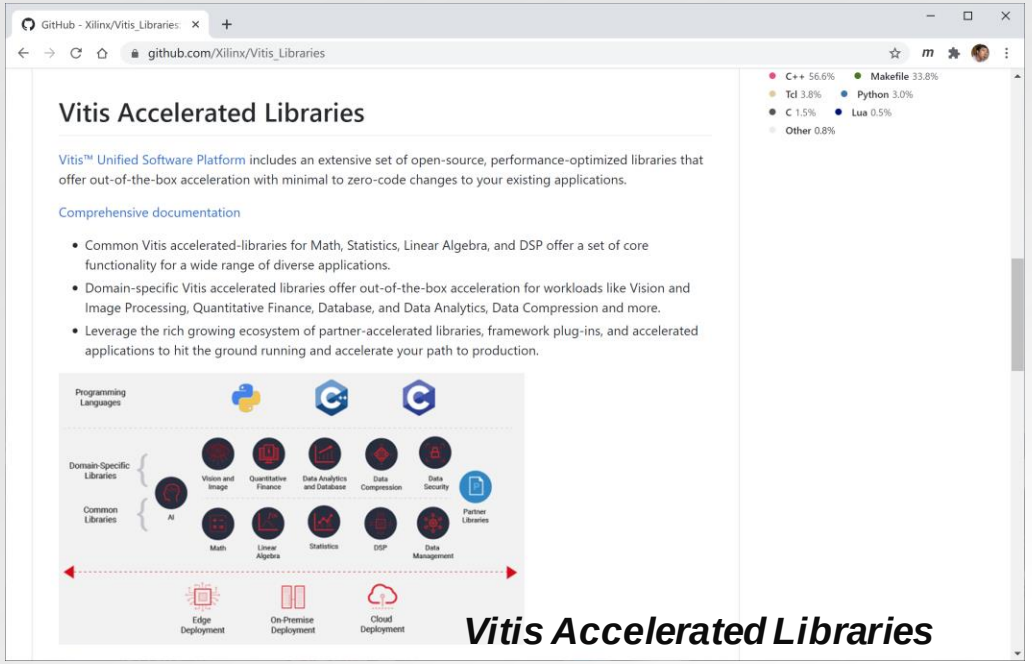


Vitis HLS and Accelerated Libraries

alpha_mix.cpp

```
103     }
104 }
105
106
107 // function to stream out the cv::Mat data
108 // input hls::stream data width is 24*8, output AXI stream data width is 64
109 void cvmat_stream_out(xf::cv::Mat<XF_8UC3, MAX_HEIGHT, MAX_WIDTH, XF_NPPC8> &input_img,
110                       hls::stream<ap_axiu<64, 0, 0, 0>> &stream_out)
111 {
112     for (int i = 0; i < input_img.rows * (input_img.cols >> XF_BITSHIFT(XF_NPP8)); i++)
113     {
114         ap_uint<24 * XF_NPPC8> data;
115         ap_axiu<64, 0, 0, 0> v;
116         data = input_img.read(i);
117         v.data = data.range(63,0);
118         stream_out.write(v);
119         v.data = data.range(127,64);
120         stream_out.write(v);
121         v.data = data.range(191,128);
122         stream_out.write(v);
123     }
124 }
125
126 // Top level kernel function
127 void alpha_mix(hls::stream<ap_axiu<64, 0, 0, 0>> &time_img_input,   // time image input
128                ap_uint<512> *bgr_img_input,                        // background image input
129                hls::stream<ap_axiu<64, 0, 0, 0>> &mix_img_output,  // mixed image output
130                int     time_img_rows_in,                           // input time image height
131                int     time_img_cols_in,                           // input time image width
132                int     time_img_rows_rsz,                          // resized time image height
133                int     time_img_cols_rsz,                          // resized time image width
134                int     time_img_pos_row,                           // resized time image position - Y
135                int     time_img_pos_col,                           // resized time image position - X
136                ap_uint<32> time_char_color,                        // [31:0] = [xRGB]
137                ap_uint<32> time_bgr_color,                         // [31:0] = [xRGB]
138                int     time_bgr_opacity,                           // time image background opacity [7:0] is used
139                int     bgr_img_rows,                               // background image height
140                int     bgr_img_cols                                // background image width
141
```

Customer HLS C Code



Vitis Accelerated Libraries

Vitis HLS

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

Vitis HLS Kernel

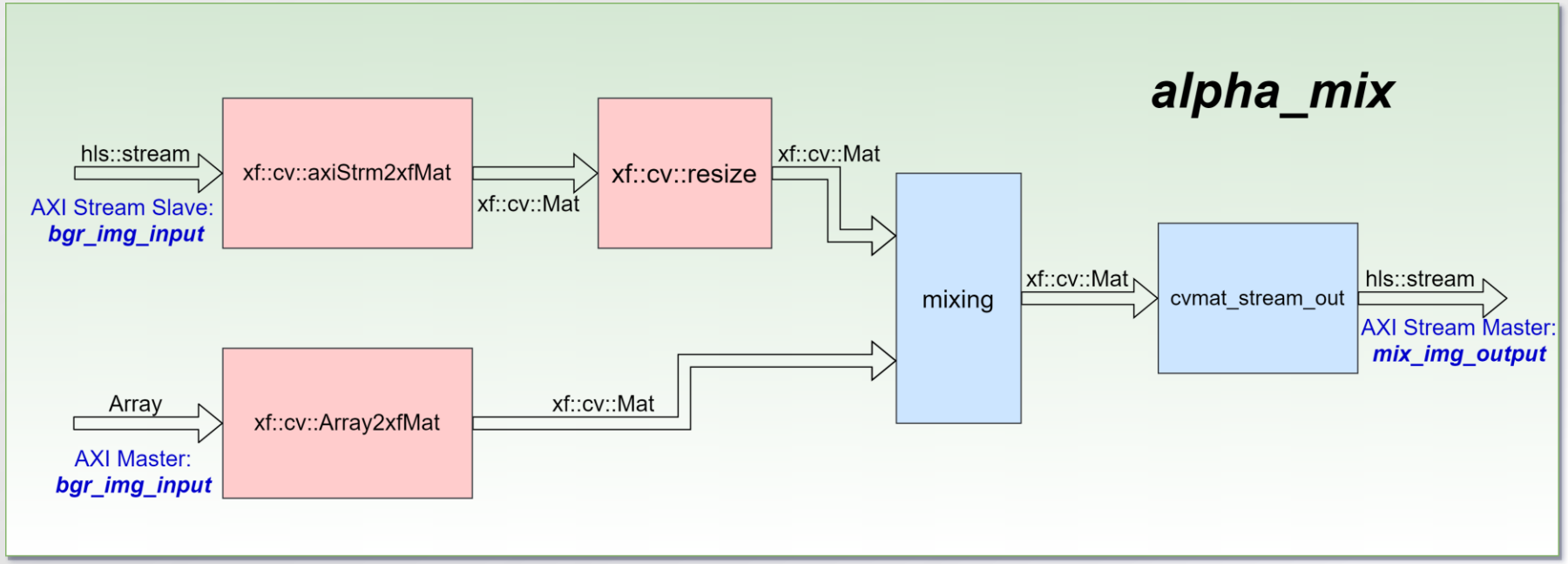
Vitis Accelerated Libraries provide a rich set of fully verified designs, which can easily used to reduce the design cycle.

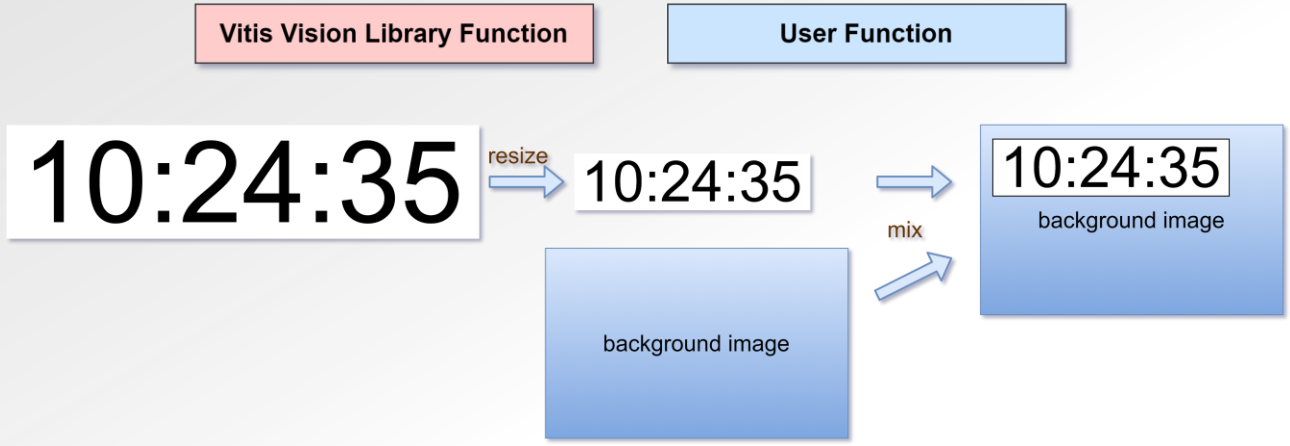
XILINX.

Example HLS C Kernel Design with Vitis Vision Library

HLS C kernel: *alpha_mix*

- *Top function*
  - *alpha_mix*
- *AXI Control Slave*
- *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





XILINX.

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.

XILINX.

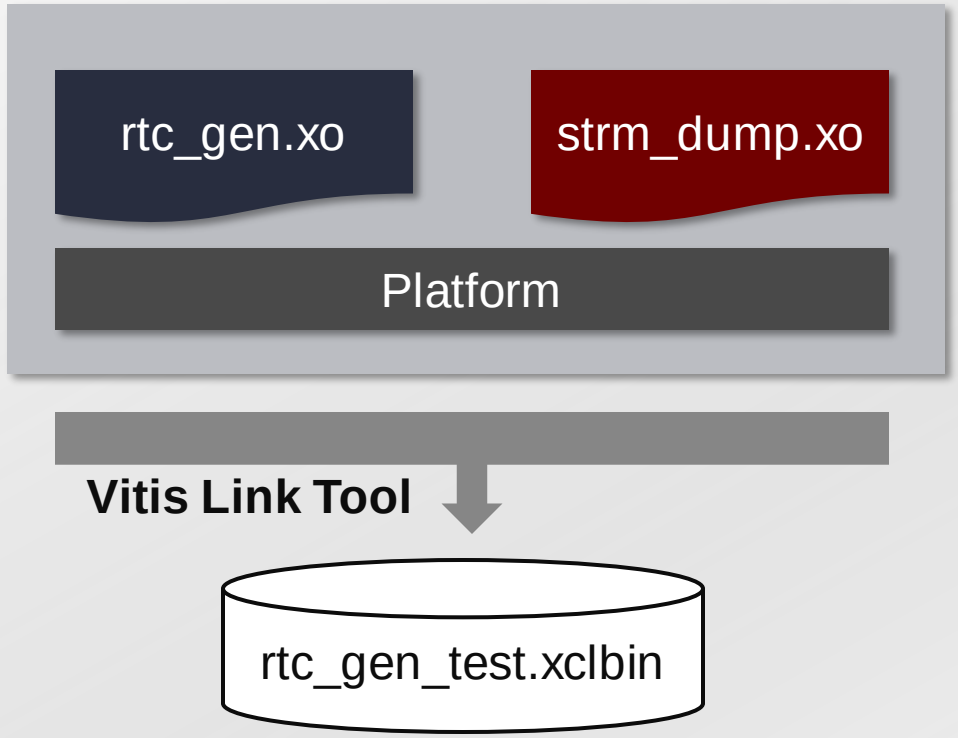
Example Simple HLS C Kernel Design

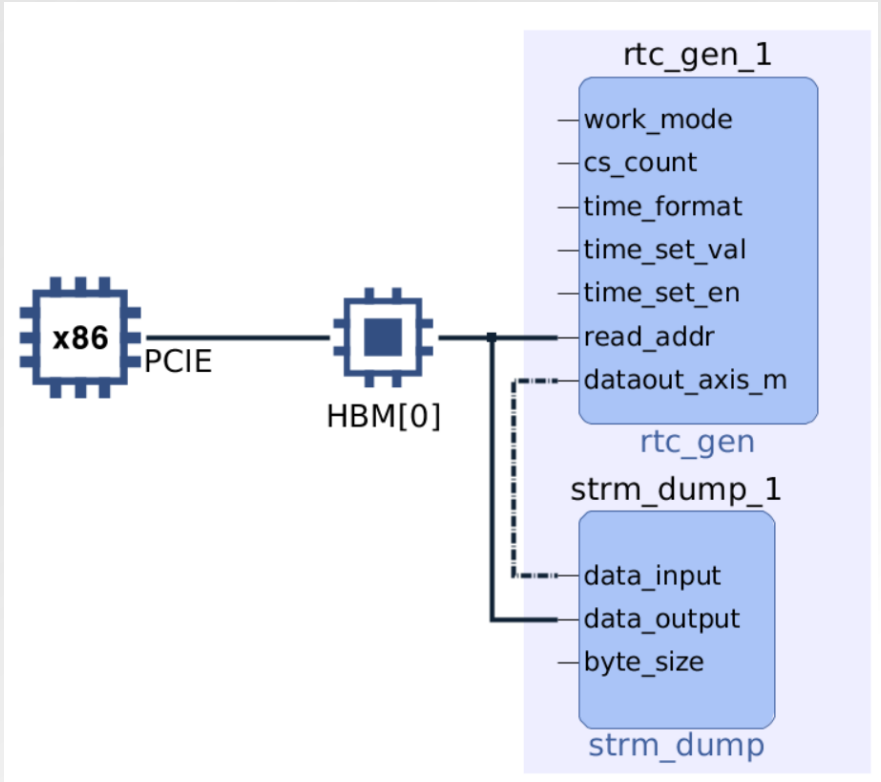
```
extern "C"
{
    // kernel function.
    void strm_dump (hls::stream<ap_axiu<INPUT_PTR_WIDTH, 0, 0, 0>>& data_input,
                    ap_uint<OUTPUT_PTR_WIDTH> *data_output,
                    int byte_size)
    {
        for (int i = 0; i < (byte_size / 8); i++) {
        // clang-format off
        #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)



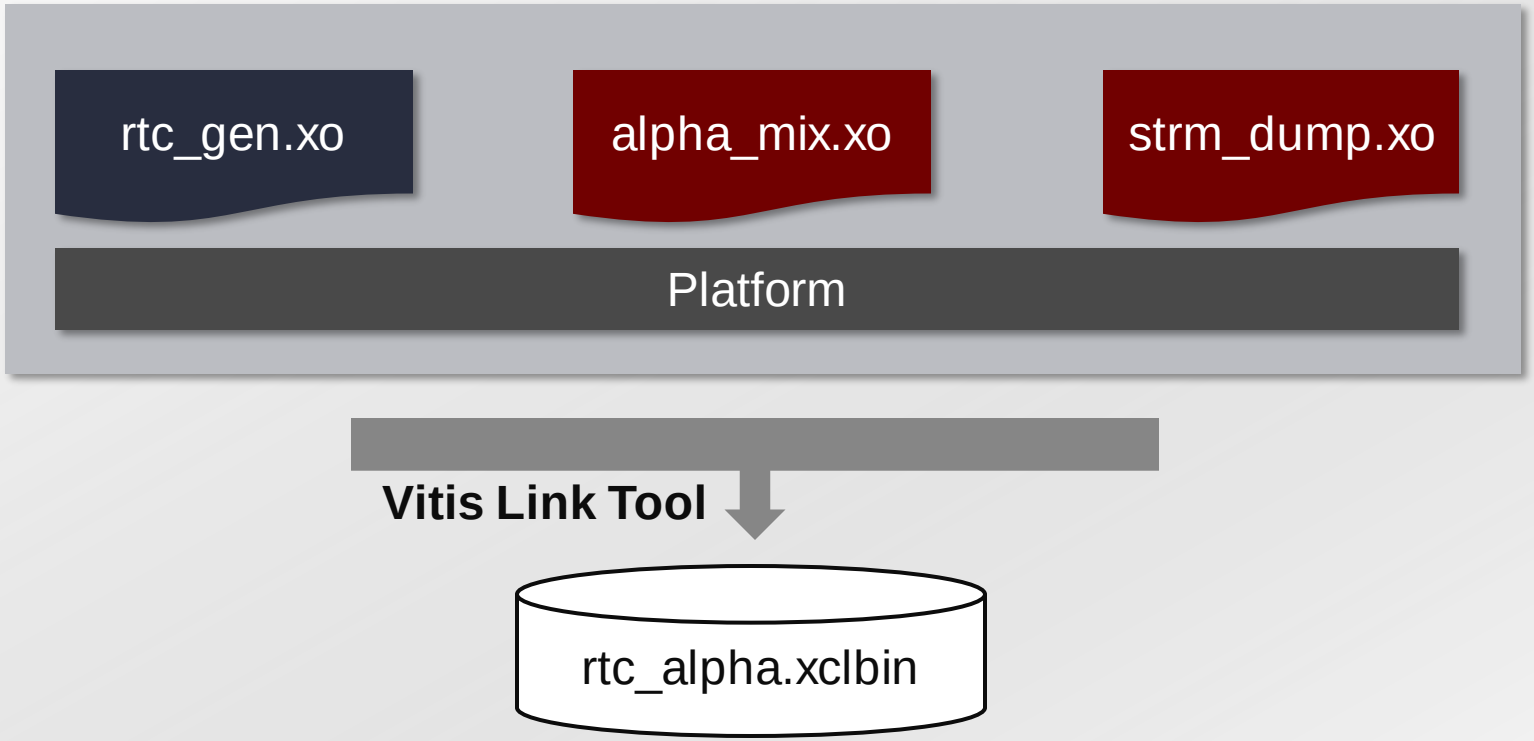


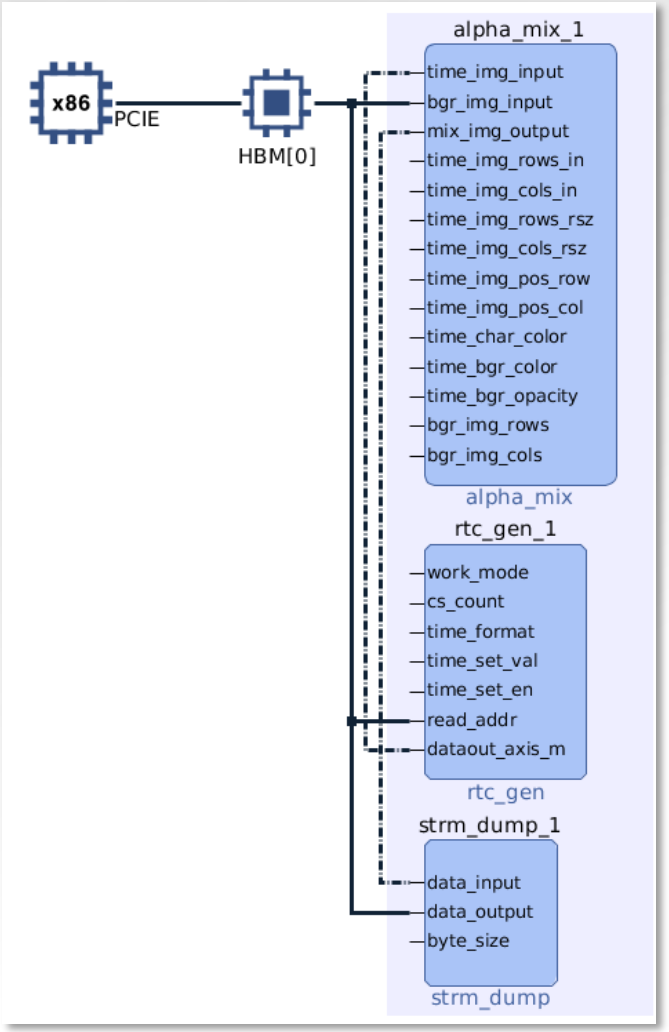

```
v++ -l                                          \
    -platform xilinx_u50_gen3x16_xdma_201920_3  \
    -config xclbin_rtc_gen_test.ini             \
    -o rtc_gen_test.xclbin                      \
    rtc_gen.xo strm_dump.xo
```

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
```

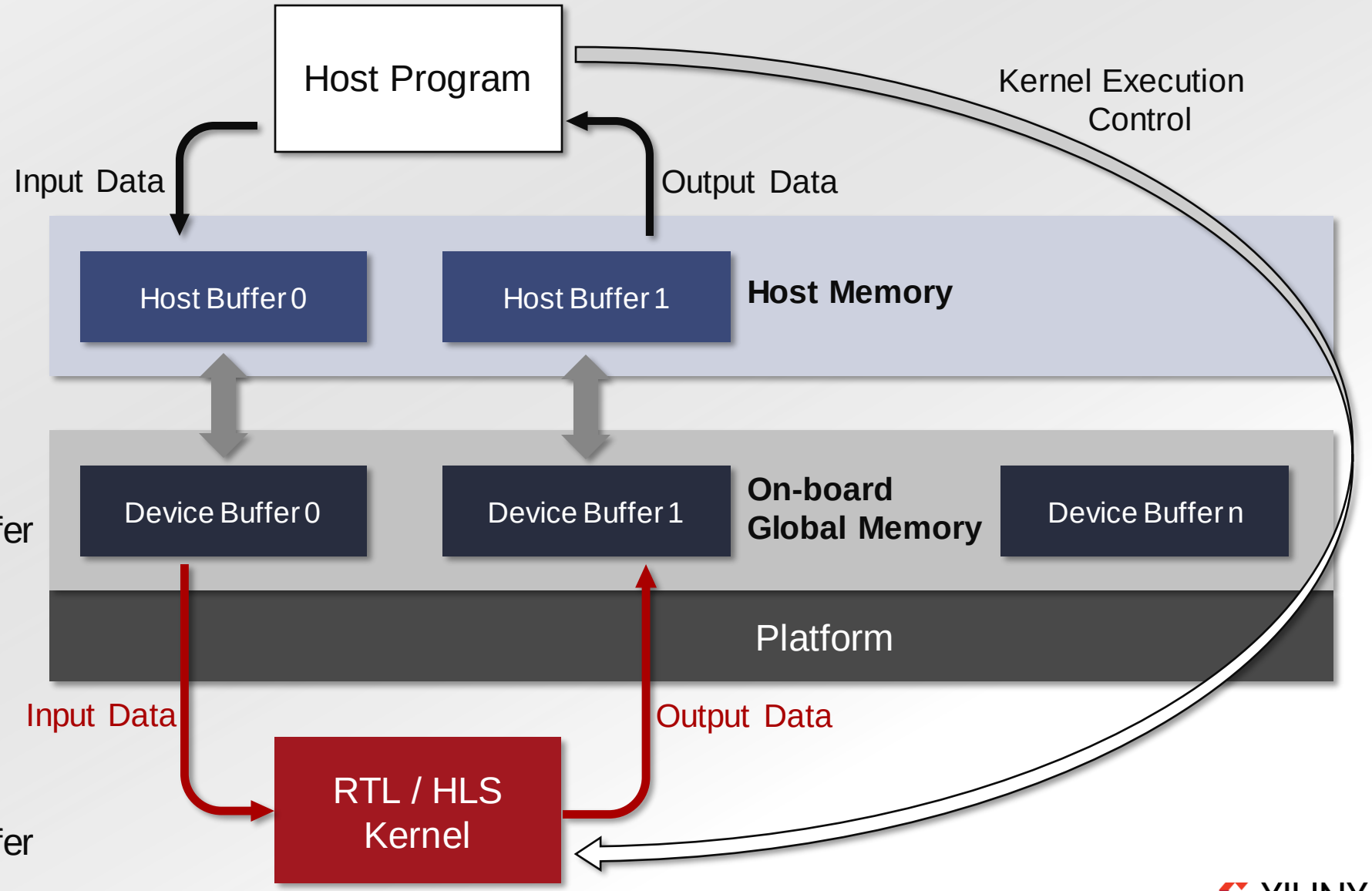
xclbin_rtc_alpha.ini

```
[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
```

XILINX.

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 outpu data from device buffer to host buffer

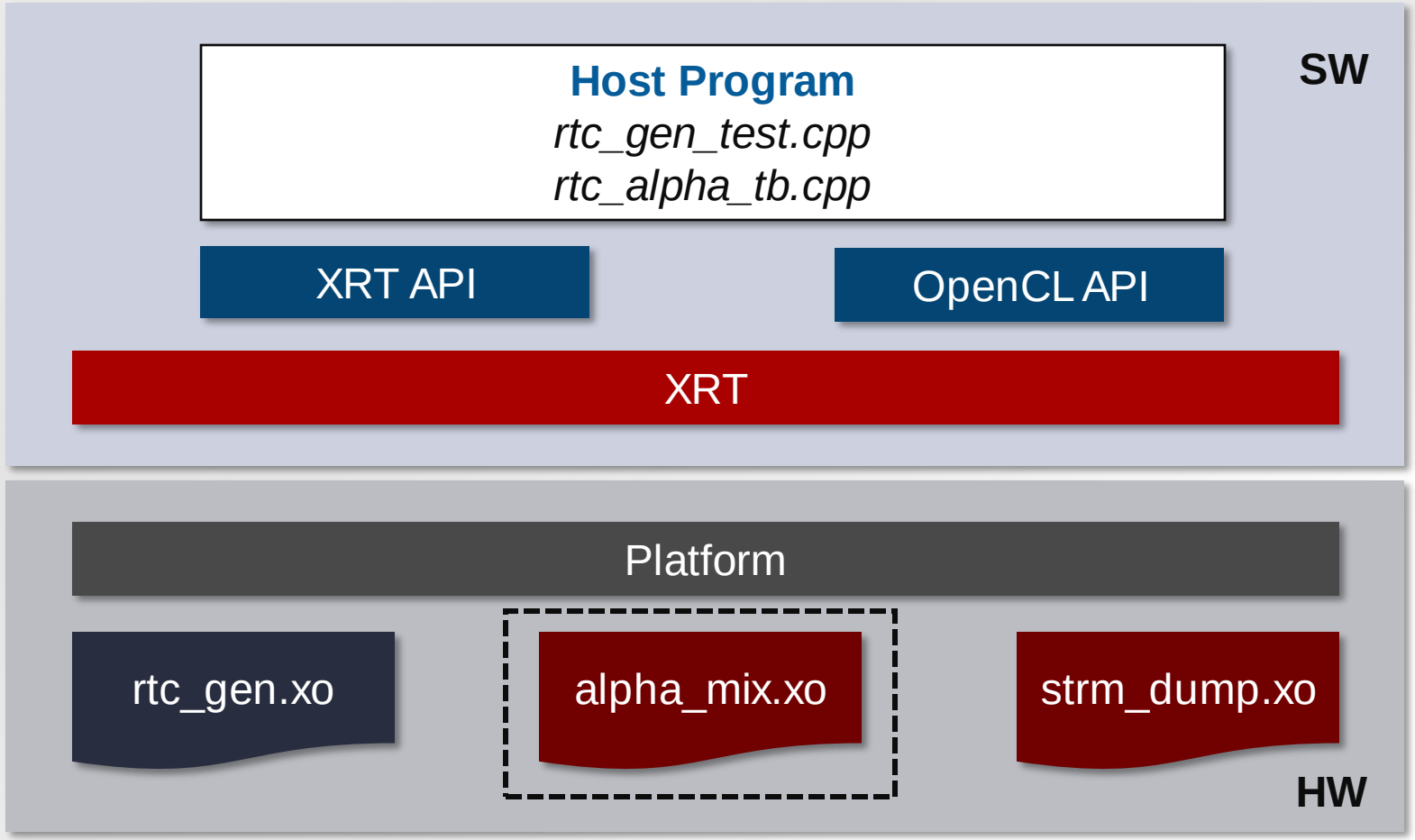


XILINX

Example Host Program with Different Level of API

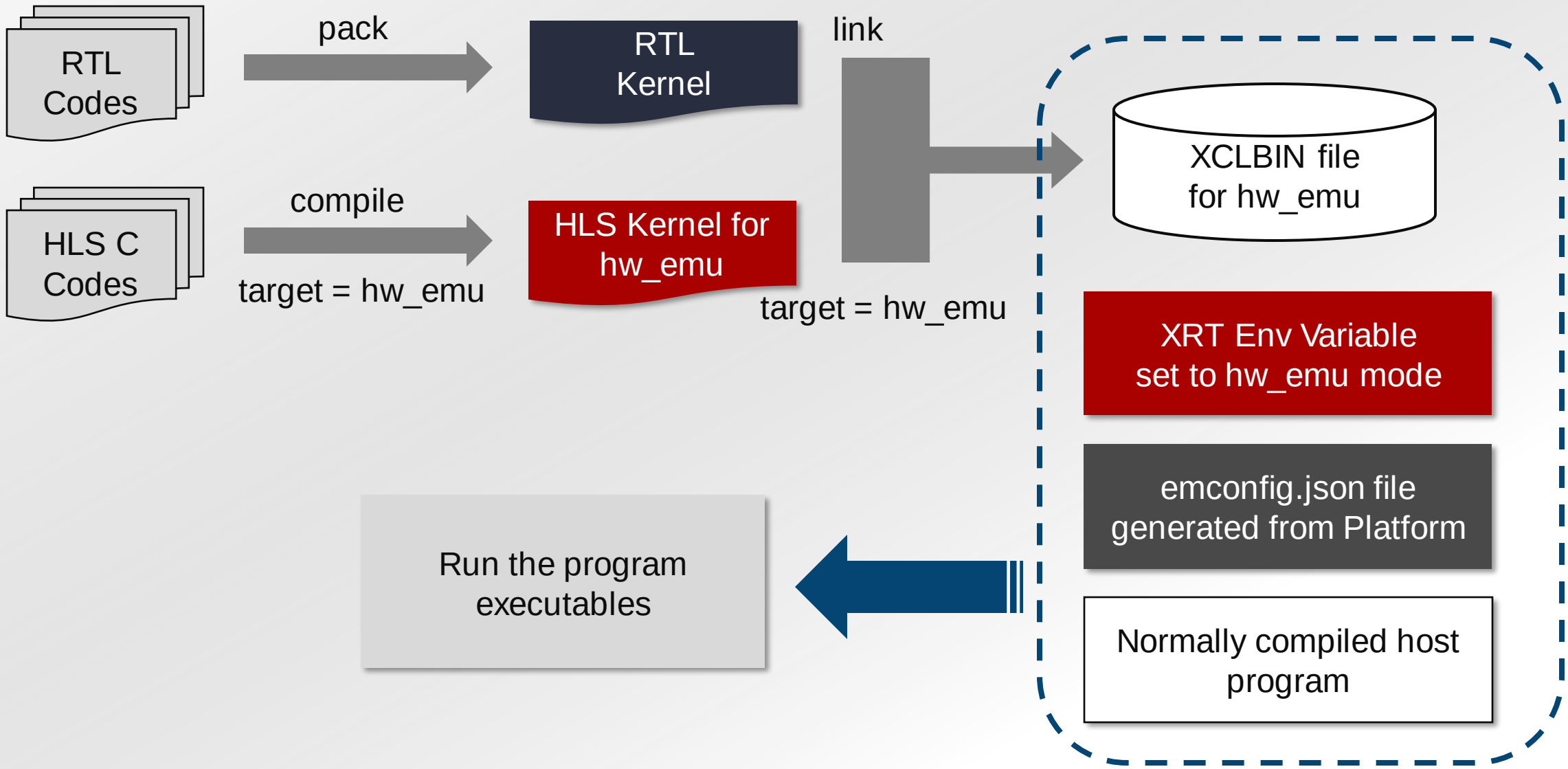
Host program:
rtc_gen_test.cpp
rtc_alpha_tb.cpp

- *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*



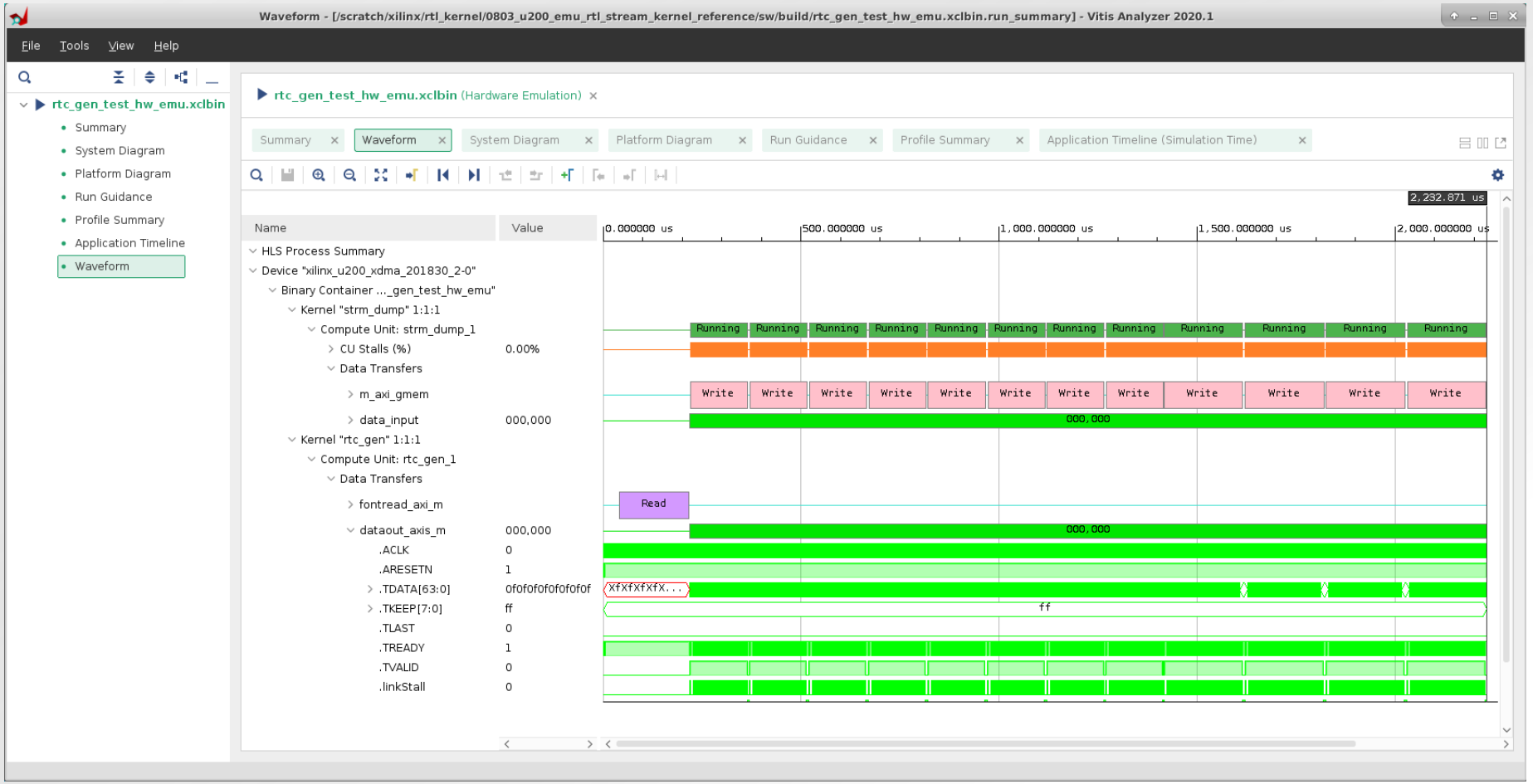
In the example host program, the usage of low-level API *xclRegWrite* / *xclRegRead* is illustrated. User can use these two functions to access the AXI slave register directly, which might be valuable for specific control schemes of some traditional RTL design.

Use Emulation to Verify Mixed Kernel Designs



Use Emulation to Verify Mixed Kernel Designs

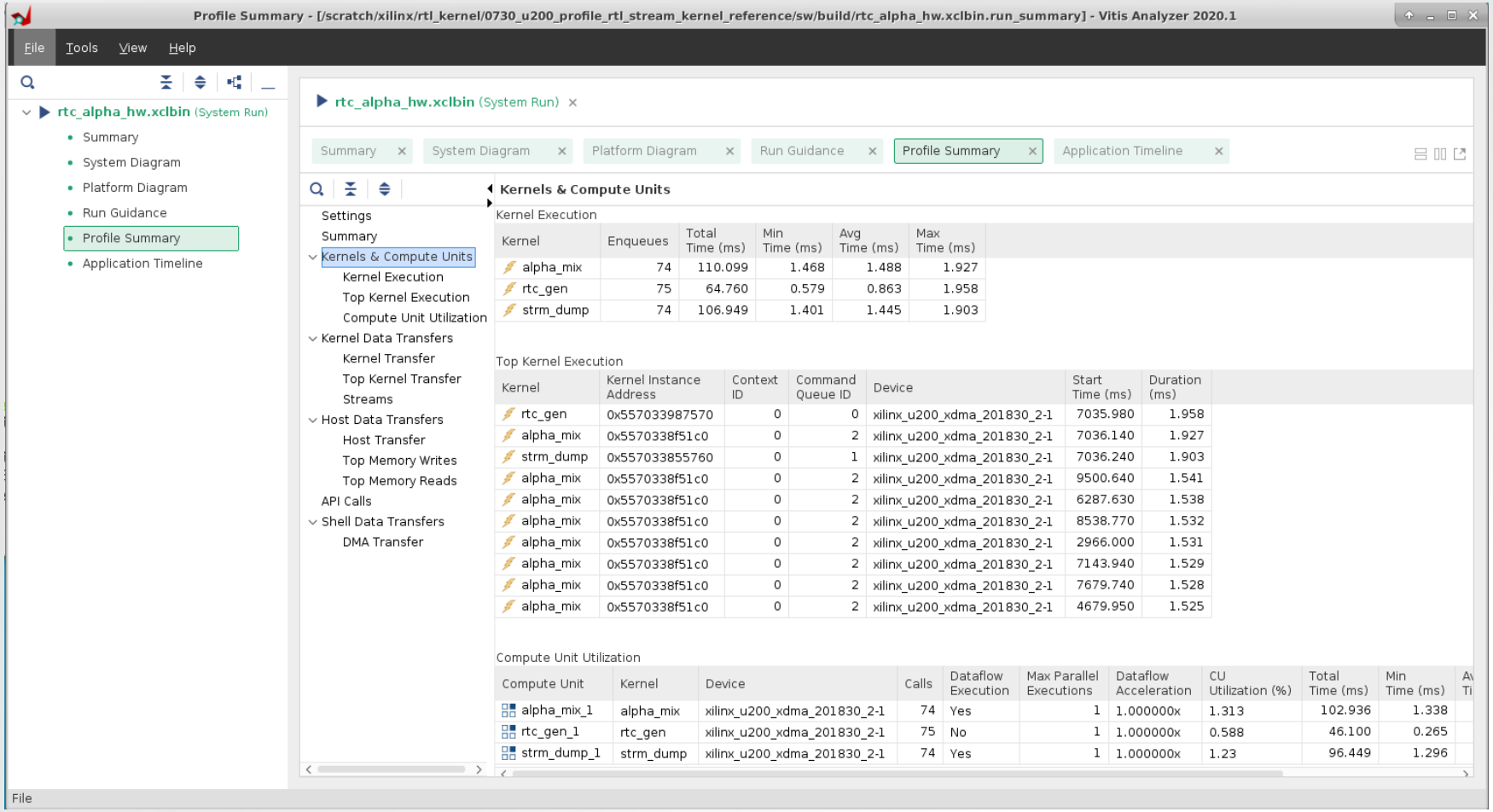
Hardware Emulation – provides insight to the interaction details of the kernels, platform and software



XILINX.

Use Vitis Analyzer to Profile the System Designs

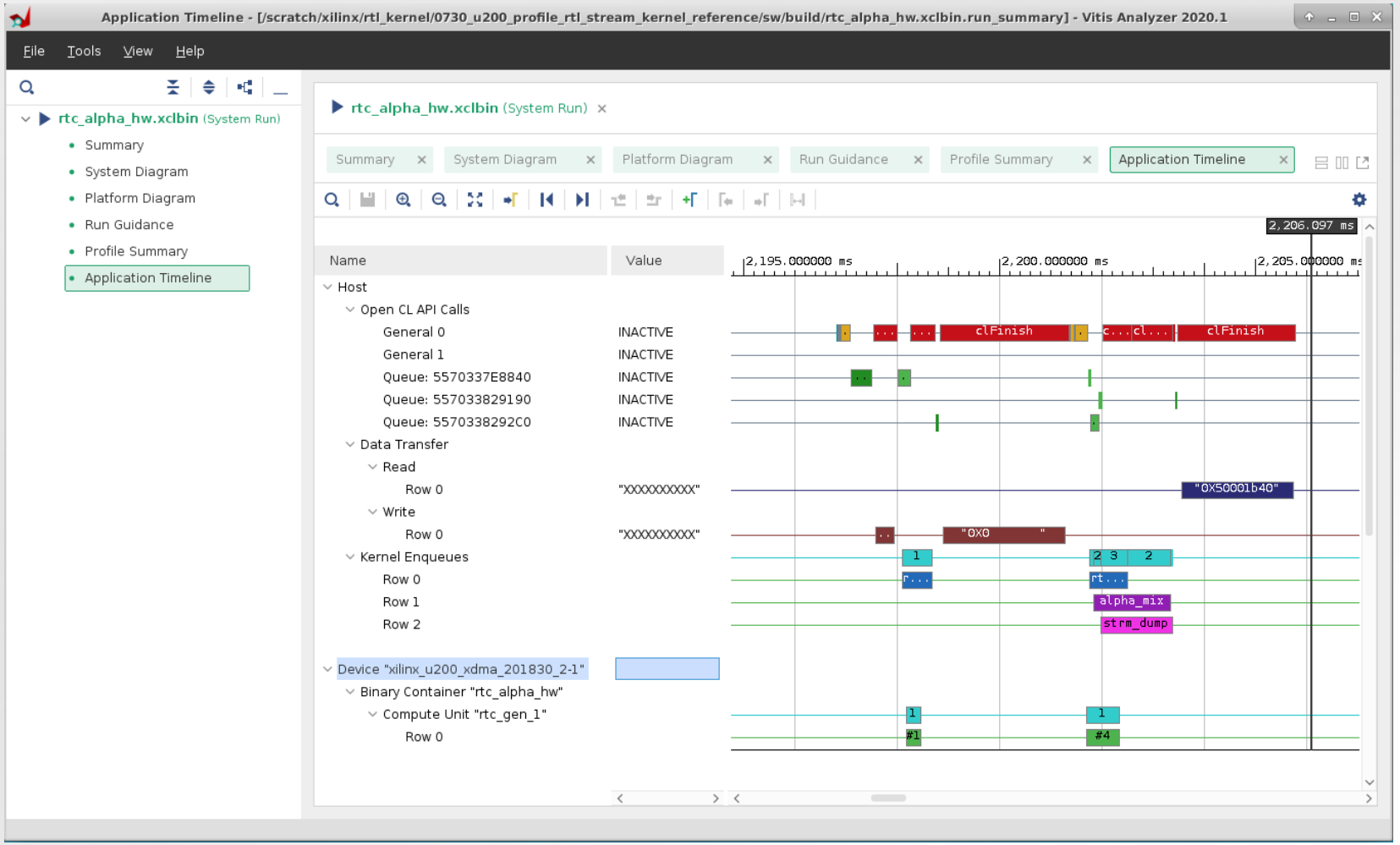
Profiling – provides inclusive statistics information for software and kernel execution

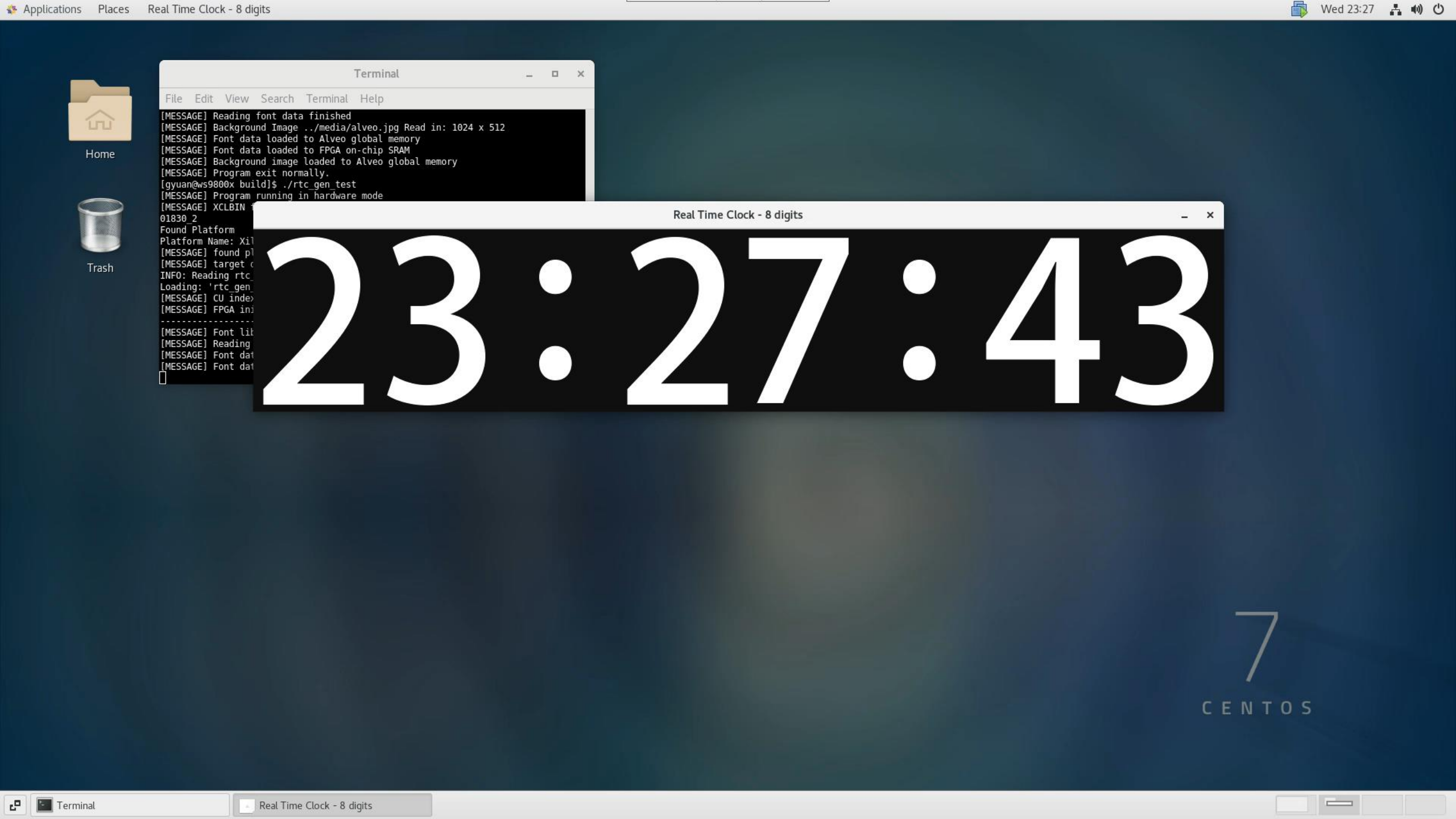


XILINX.

Use Vitis Analyzer to Profile the System Designs

Profiling – provides inclusive statistics information for software and kernel execution




Applications
Places
Real Time Clock - 8 digits
Wed 23:27
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
Real Time Clock - 8 digits
23:27:43
7
CENTOS
Terminal
Real Time Clock - 8 digits

XILINX

Thank you!