**[Application Acceleration with High-Level Synthesis](https://github.com/TimHsu28/Tim-Hsu)**

**[Lab A No.11\_Streaming\_free\_running\_k2k](https://github.com/TimHsu28/Tim-Hsu)**

110061585 Tim Hsu

Streaming\_free\_running\_k2k

* The free-running kernel has no memory input or output port, and therefore it interacts with the host or other kernels (other kernels can be regular kernel or another free-running kernel) only through streams.
* When the FPGA is programmed by the binary container (xclbin), the free-running kernel starts running on the FPGA, and therefore it does not need to be started from the host code.
* The kernel works on the stream data as soon as it starts receiving from the platform I/O or other kernels, and it stalls when the data is not available.

Kernel to kernel streaming example consisting of three compute units in a linear hardware pipeline.  
    1) Memory read

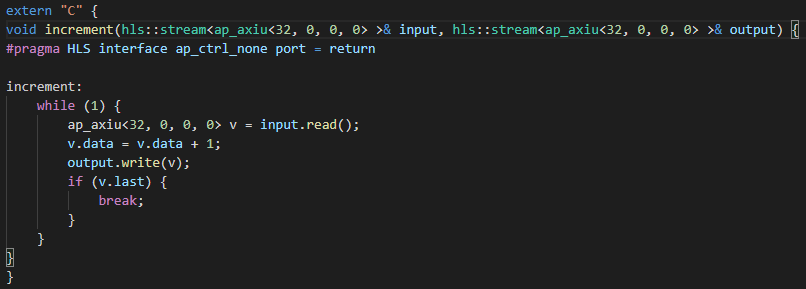
       This Kernel reads the input vector from Global Memory and streams its output.  
    2) Increment

      This Kernel reads stream input, increments the value and streams to output.  
    3) Memory write

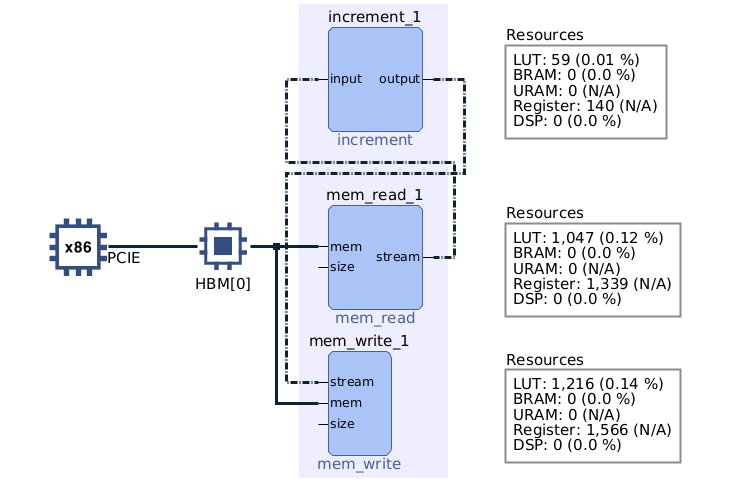
        This Kernel reads from its input stream and writes into Global Memory

For free running kernel, user needs to specify ap\_ctrl\_none for return port.

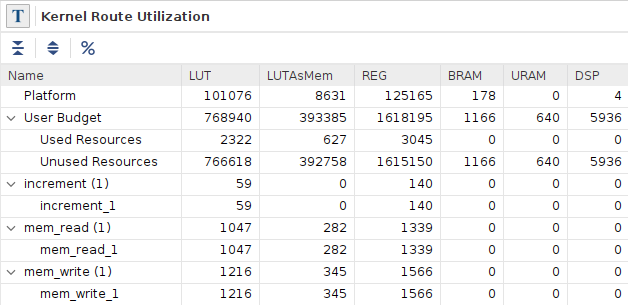
This will create the kernel without AXI lite interface. Kernel will always be in running states



Example of system diagram.



Example of resource utilization.

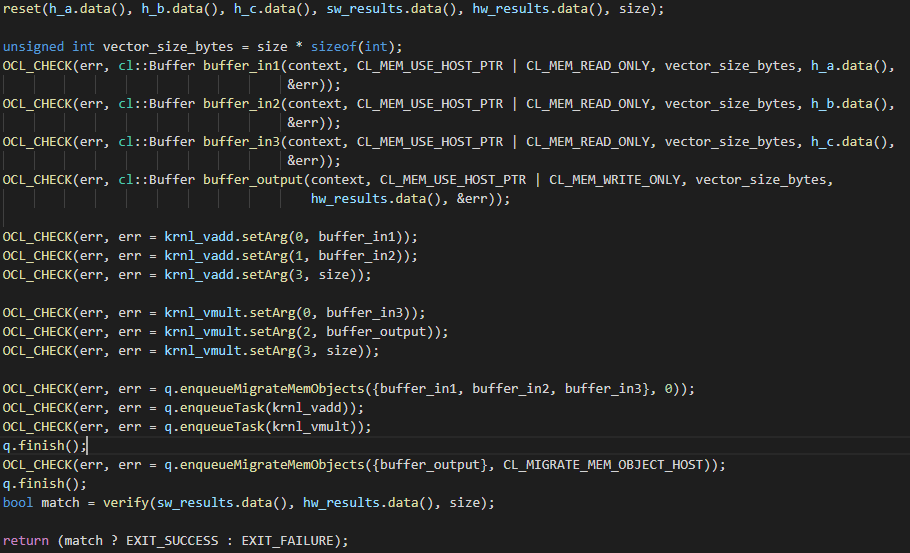


Streaming\_k2k\_mm

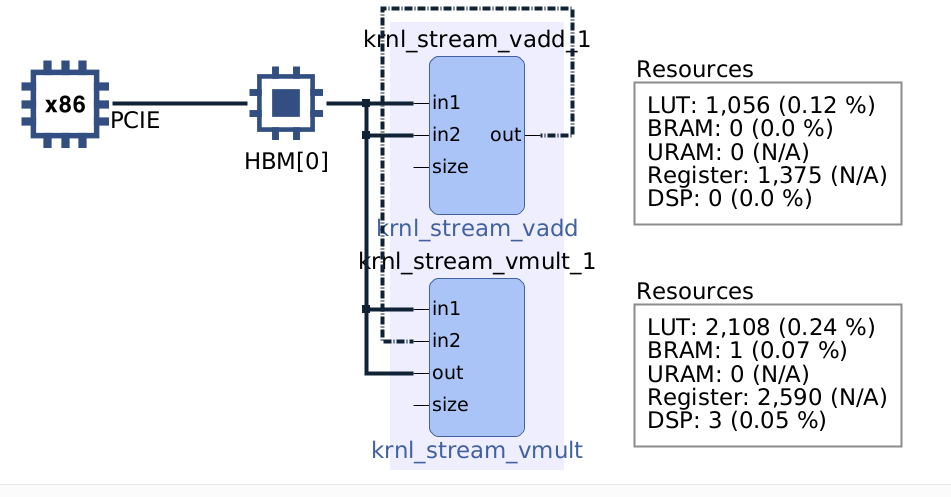
the free-running kernel only contains hls::stream inputs and outputs. The recommended coding guidelines include:

* Use hls::stream<ap\_axiu<D,0,0,0> > if the port is interacting with another stream port from the kernel.
* Use hls::stream<qdma\_axis<D,0,0,0> > if the port is interacting with the host.
* Use the hls::stream data type for the function parameter causes Vitis HLS to infer an AXI4-Stream port (axis) for the interface.
* The free-running kernel must also specify the following special INTERFACE pragma

On the code below it reset the data vectors and run the kernel after it allocate buffer in global memory. Moreover, Buffers are allocated CL\_MEM\_USE\_HOST\_PTR for efficient memory and Device-to-host communication. When the Kernel Arguments have been set, it also copy input data to device global memory for launching the Kernel after that it copy Result from Device Global Memory to Host Local Memory then open CL Host Code Ends to compare the device results with software results.



Example of system diagram.



Example of resource utilization.

