Chapter 4 in "**UG1399 (v2022.2) December 7, 2022" explains the array in HLS**

AMD
XILINX

Chapter 4

Arrays Primer

Mapping Software Arrays to Hardware Memory

Arrays are a fundamental data structure in any C++ software program. Software programmers view arrays as simply a container and allocate/deallocate arrays on demand - often dynamically. This type of dynamic memory allocation for arrays is not supported when the same program needs to be synthesized for hardware. For synthesizing arrays to hardware, knowing the exact amount of memory (statically) required for your algorithm becomes necessary. In addition, the memory architecture on FPGAs (also called "local memory") has very different trade-offs when compared to global memory which is often the DDR or HBM memory banks. Access to global memory has high latency costs and can take many cycles while access to local memory is often quick and only takes one or more cycles.

When an HLS design has been suitably pipelined and/or unrolled, the memory access pattern becomes established. Vitis HLS allows users to map arrays to various types of resources - where the array elements are available in parallel with or without handshaking signals. Both internal arrays and arrays in the top-level function's interface can be mapped to registers or memories. If the array is in the top-level interface, Vitis HLS automatically creates the address, data, and control signals required to interface to external memory. If the array is internal to the design, Vitis HLS not only creates the necessary address, data, and control signals to access the memory but also instantiates the memory model (which is then inferred as memory by the downstream RTL synthesis tool).

Arrays are typically implemented as memory (RAM, ROM, or shift registers) after synthesis. Arrays can also be fully partitioned into individual registers to create a fully parallel implementation provided the platform has enough registers to support this step. The initialization_and_reset example available on GitHub demonstrates different implementations of memory.

Arrays on the top-level function interface are synthesized as RTL ports that access external memory. Internal to the design, arrays sized less than 1024 will be synthesized as a shift register. Arrays sized greater than 1024 will be synthesized into block RAM (BRAM), LUTRAM, or UltraRAM (URAM) depending on the optimization settings (*see BIND_STORAGE directive/pragma*).