SYCL for Xilinx Versal ACAP AIE CGRA
Keryell, Ronan; Gozillon, Andrew; Harnisch, Gauthier; Kwon, Hyun; Chakaravarthy, Ravikumar; Wittig, Ralph

Accepted/In press: 27/04/2021

Document Version
Peer reviewed version

Link to publication on the UWS Academic Portal

Citation for published version (APA):
Keryell, R., Gozillon, A., Harnisch, G., Kwon, H., Chakaravarthy, R., & Wittig, R. (Accepted/In press). SYCL for Xilinx Versal ACAP AIE CGRA. Poster session presented at IWOCL SYCLCon 2021,

General rights
Copyright and moral rights for the publications made accessible in the UWS Academic Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact pure@uws.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
SYCL for Xilinx Versal ACAP AIE CGRA
Keryell, Ronan; Gozillon, Andrew; Harnisch, Gauthier; Kwon, Hyun; Chakaravarthy, Ravikumar; Wittig, Ralph

Accepted/In press: 27/04/2021

Document Version
Early version, also known as pre-print

Link to publication on the UWS Academic Portal

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the UWS Academic Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact pure@uws.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
SYCL for Xilinx Versal ACAP AIE CGRA
Andrew Gozillon\textsuperscript{2} Gauthier Harnisch\textsuperscript{1} Ronan Keryell\textsuperscript{1} Hyun Kwon\textsuperscript{1} Ravikumar Chakaravarthy\textsuperscript{1} Ralph Wittig\textsuperscript{2}

\textsuperscript{1}Xilinx \hspace{1em} \textsuperscript{2}University of the West of Scotland

\section*{Abstract}
SYCL is a single-source C++ DSI targeting a large variety of accelerators in a unified way by using different backends. Xilinx Versal ACAP is a new system-on-chip (SoC) device integrating various computing resources like various CPUs, an FPGA, a coarse-grain reconfigurable array (CGRA), etc. interconnected by different network-on-chip (NoC).

The AIE CGRA is an array of 400 VLIW DSP operating on 512-bit vectors with their own neighborhood distributed memory (32 KB data, 16 KB instruction).

We expose architectural details to the programmer through some SYCL extensions and extend SYCL with a geographical collective model.

The SYCL implementation targeting the AIE CGRA by merging 2 different open-source implementations, Intel’s omniDPC++ with some LLVM passes from triSYCL and a new SYCL runtime from triSYCL.

The SYCL device compiler generates LLVM IR for the Synopsys ASIP CHESS compiler generating the AIE instructions.

The host runtime runs on the ARM A72 CPU of the ACAP and controls the CGRA through the Xilinx liibssing=io-v2 library.

\section*{Weaving heterogeneous tiles and memories with meta-programming}

\begin{itemize}
  \item First approach: expose geographical view of the chip
  \begin{itemize}
    \item Each tile is a sub-device with some neighborhood
    \item Can launch tasks on each tile
    \item Can use AIE-specific functions inside kernels
    \item Neighbor communications
    \item DMA
    \item Hardware locks
    \item Control of the AIE NoC done by host
    \item Close to usual SYCL programming model otherwise
  \end{itemize}

  \item Second approach: entangled mode by weaving tiles and memories
  \begin{itemize}
    \item Plan SYCL can be cumbersome with 400 tiles to program...
    \item Provide a cooperative mode allowing meta-programming tiles and memories
    \item Compile-time specialization of tiles (instructions) and memories (data)
    \item Allow each tile to access neighbor tile memories in a type-safe way
  \end{itemize}
\end{itemize}

\section*{Implementation}

\begin{itemize}
  \item Full SYCL compiler & runtime implementation
  \item Run on real hardware or hardware simulator
  \item Pure SYCL C++ implementation
  \item No specific compiler required
  \item Run on (laptop) host CPU at full C++ speed, standard debugging, thread-simulator of hardware features...
    \begin{itemize}
      \item 1 thread per host...
      \item 1 thread per AIE tile
      \item 1 thread per GPU work-item
      \item 1 thread per FPGA work-item
    \end{itemize}
  \item Easy code instrumentation for statistics by adapting SYCL C++ classes
  \item Use normal debugger
  \item Glo is writable in Python to expose new features
  \item Can experiment with Xilinx devices from year 2030
  \item Mix-and-match
  \item Run some parts of the hardware remotely or in simulators
  \item Allows kernels on host CPU while using memory-mapped real hardware (DMA, AXI streams, NoC…)
  \item Distribute execution across data-center (Colbrity SYCL for MPI-SYCL)
\end{itemize}

\section*{Conclusion}

\begin{itemize}
  \item Provide direct programming in C/C++ and SYCL for CGRA like Xilinx ACAP VC1902
  \item Complement existing Xilinx ML environment and Xilinx ADF for Kahn’s Process Network
  \item Follow SYCL philosophy: based on plain C++ without extensions
  \item Implementation based on triSYCL, omniDPC++ and Synopsys ASIP Designer chessa-clang
  \item Single-source pure C++ is incredibly powerful for hardware-software co-design
  \item Provide CPU emulation for architecture research and co-design
  \item Can detect hardware concurrency issues with CPU emulation
  \item More info can be found and merged into https://github.com/triSYCL/triSYCL
  \item Retargetable to other CGRAs
\end{itemize}