Improving Memory Interfacing in HLS-Generated Accelerators with Custom Caches

Claudio Barone, Giovanni Gozzi, Michele Fiorito,
Ankur Limaye, Antonino Tumeo, Fabrizio Ferrandi

SC23 Denver, CO | i am hpc.

Motivations

- **Domain Specific Accelerators** are becoming more popular and are widely used in datacenters and cloud computing
- **High-Level Synthesis (HLS)** tools can help in creating custom accelerators
- Commercial HLS tools focus their optimization efforts on the computation, **leaving memory transfers behind**
- Optimizing memory transfers with commercial tools requires **significant code restructuring**

User effort and possible customization

- Adding caches to an accelerator requires **no changes to internal code**
- **Parametric caches** offer great opportunities to the user by selecting the most appropriate configuration (sizes, write policy, associativity…)
- User can add **multiple caches** to the same accelerator with **different configurations**
- User controls which caches are **shared or private**
- Adding caches require **low user effort**

```
48  #pragma HLS_interface a m_axi direct bundle = gmem0
49  #pragma HLS_interface b m_axi direct bundle = gmem1
50  #pragma HLS_interface output m_axi direct bundle = gmem2
51
52  #pragma HLS_cache bundle = gmem0 way_size = 16 line_size = 16
53  #pragma HLS_cache bundle = gmem1 way_size = 16 line_size = 16
54  #pragma HLS_cache bundle = gmem2 way_size = 16 line_size = 16
55  void mmult(int* a, int* b, int* output)
```

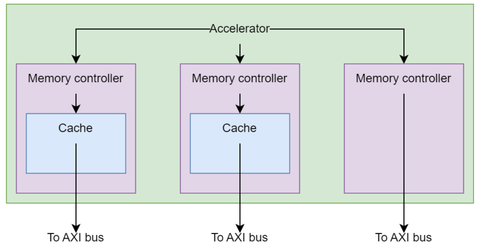
Cache declarations

Contributions

- New methodology to **reduce impact of memory latency** for custom accelerators
- **Improvements** to IObundle (IOb) [1] caches
- **Integration** in open-source Bambu [2] HLS tool
- **Evaluation** of the impact of custom caches in terms of **execution time and resource utilization**

Architecture and memory operations

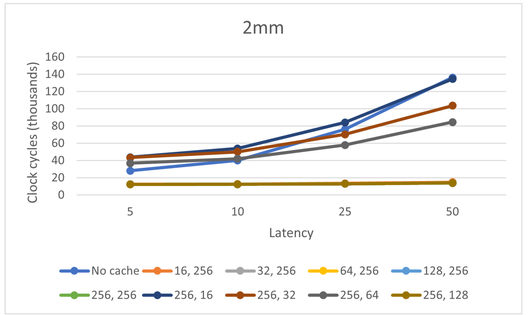
- The HLS tool inserts caches inside the standard memory controller
- To rest of the accelerator, as well as the rest of the system, their insertion is completely transparent
- Fully take advantage of AXI [3] bursts
- Bursting allows reading multiple data with a single transaction and helps in hiding memory latency
- Support for outstanding write transactions
- Automatic cache flush at the end of the computation



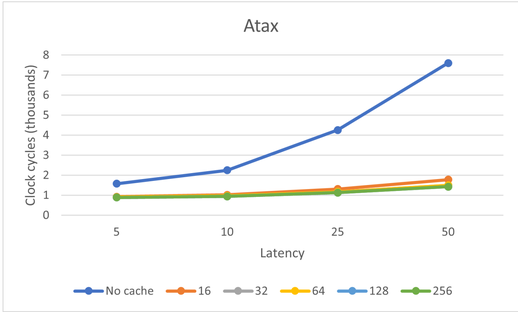
Schematic of an accelerator with three memory channels (two with caches, one cacheless)

Evaluation

- We evaluate our approach on 5 different kernels form the PolyBench [4] suite
- We consider the impact of many cache sizes at different memory latencies
- Caches shared among matrices if there is no line contention
- Separate cache size exploration depending on access pattern



Execution delay in clock cycles - 2mm



Execution delay in clock cycles - Atax

- Experimental results show the potential of our approach to HLS caches
- Very positive results at high memory latency
- Possible penalty at low latencies, depending on cache size and matrix access pattern
- Similar results for doitgen, mvt, bicg kernels

Resource overhead and speedup

	No cache			16, 256			32, 256			64, 256			128, 256			256, 256			256, 16			256, 32			256, 64			256, 128		
	L	R	C	L	R	C	L	R	C	L	R	C	L	R	C	L	R	C	L	R	C	L	R	C	L	R	C	L	R	C
2mm	6671	6629	136161	1.36	1.19	9.1	1.44	1.27	9.75	1.45	1.27	9.75	1.44	1.28	9.75	1.42	1.27	9.75	1.36	1.24	1.01	1.42	1.28	1.31	1.42	1.27	1.61	1.43	1.27	9.75
doitgen	4520	4901	739564	1.36	1.18	9.62	1.49	1.23	10.1	1.45	1.23	10.1	1.46	1.23	10.1	1.44	1.23	10.1	1.32	1.17	1.07	1.46	1.23	1.36	1.45	1.23	1.56	1.45	1.23	10.1

Resource utilization overhead (for registers, LUTs - R,L) and speed up (as execution delay in clock cycles - C) with 50 clock cycles of memory latency - 2mm and doitgen

	No cache			16			32			64			128			256		
	L	R	C	L	R	C	L	R	C	L	R	C	L	R	C	L	R	C
atax	7171	6393	7606	1.05	1.03	4.27	1.11	1.07	4.79	1.11	1.07	5.08	1.11	1.07	5.33	1.11	1.07	5.33
bicg	8094	6855	7332	1.06	1.04	2.97	1.11	1.07	3.32	1.11	1.07	3.94	1.24	1.07	4.09	1.11	1.07	6.37
mvt	4787	5513	14680	1.07	1.06	1.48	1.16	1.10	1.90	1.17	1.10	2.27	1.16	1.10	4.10	1.16	1.10	7.36

Resource utilization overhead (for registers, LUTs - R,L) and speed up (as execution delay in clock cycles - C) with 50 clock cycles of memory latency - atax, bicg, mvt

- Limited impact on resource utilization, at least for small caches
- Great maximum speedup

Conclusions

- We proposed a new methodology to reduce the impact of memory latency on the performance of custom accelerators
- We integrated customizable caches in a state-of-the-art, open-source, HLS tool
- Caches are flexible and can be adapted to many use cases
- High performance gain with no code restructuring and little user effort
  - Evaluation shows improvements up to 10x

References

[1] Mário P. Véstias João V. Roque, João D. Lopes and José T. de Sousa. 2021. IObCache: A High-Performance Configurable Open-Source Cache. Algorithms (July 2021). https://doi.org/10.3390/a14080218
[2] Fabrizio Ferrandi, Vito Giovanni Castellana, Serena Curzel, Pietro Fezzardi, Michele Fiorito, Marco Lattuada, Marco Minutoli, Christian Pilato, and Antonino Tumeo. 2021. Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications. In Proceedings of the ACM/IEEE Design Automation Conference (DAC'21). 1327–1330. https://doi.org/10.1109/DAC18074. 2021.9586110
[3] 2021. AMBA AXI and ACE Protocol Specification. Technical Report. ARM. https: //developer.arm.com/documentation/ihi0022
[4] Louis-Noël Pouchet and Tomofumi Yuki. 2021. Polybench/C 4.2.1. https://web. cse.ohio-state.edu/~pouchet.2/software/polybench

U.S. DEPARTMENT OF ENERGY | BATTELLE
PNNL is operated by Battelle for the U.S. Department of Energy

For additional information, contact:
Claudio Barone
claudio.barone@pnnl.gov
https://github.com/ferrandi/PandA-bambu

Acknowledgements
Research partially supported by SODA-ULTRA, a project in the Adaptive Tunability for Synthesis and Control via Autonomous Learning on Edge (AT SCALE) Laboratory Directed Research and Development (LDRD) Initiative.

9/30/2023 | PNNL-SA-189014

POLITECNICO MILANO 1863

Pacific Northwest NATIONAL LABORATORY
www.pnnl.gov