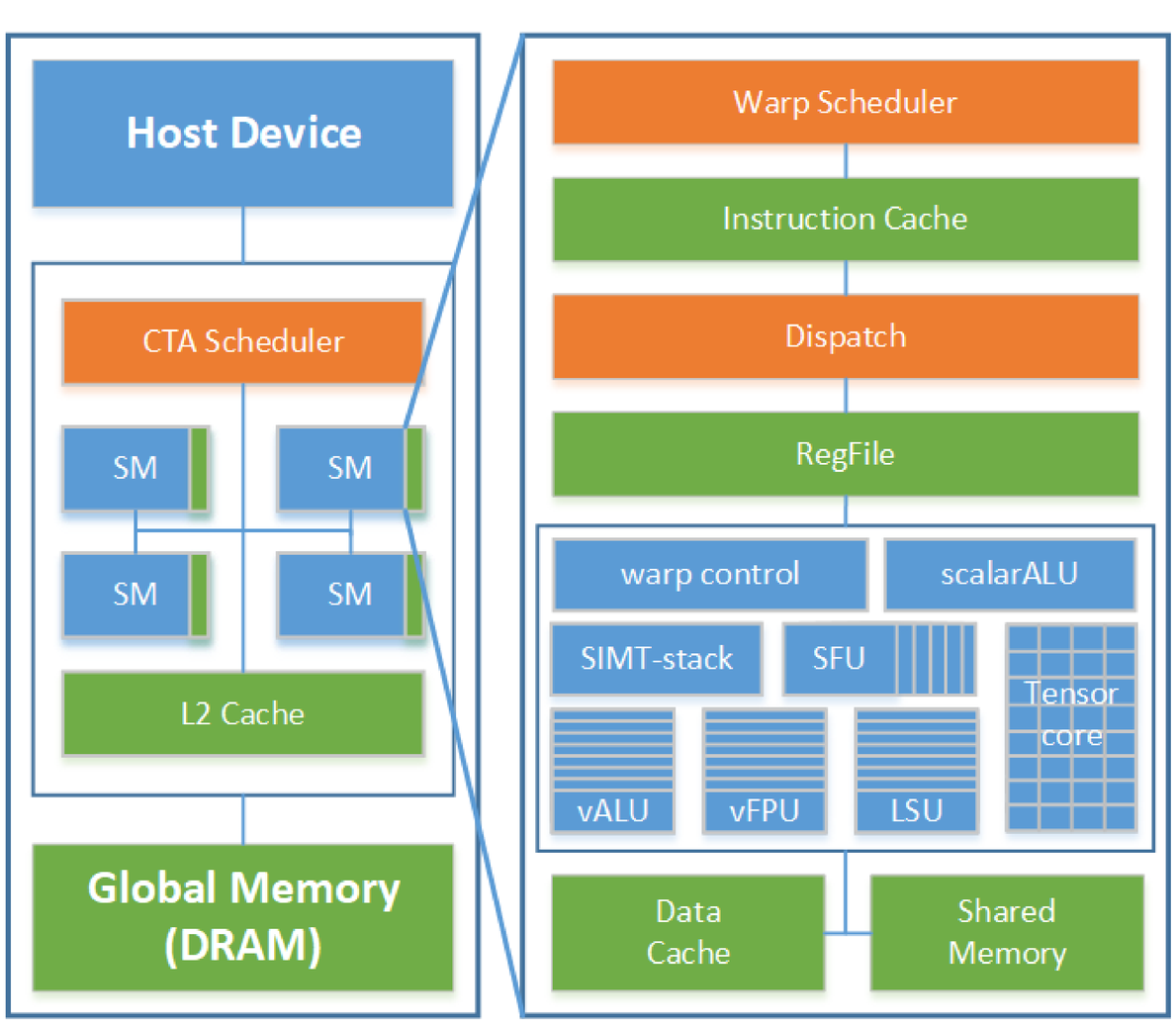
Ventus: an RVV-based General Purpose GPU Design and Implementation

Kexiang Yang[1,2], Hualin Wu[3], Jingzhou Li[1,2], Chufeng Jin[1,2], Yujie Shi[1,2], Xudong Liu[1,2],
Zexia Yang[1,2], Fangfei Yu[1,2], Mingyuan Ma[1,2], Sipeng Hu[4], Tianwei Gong[4], Hu He[1,2*]

[1]Tsinghua University, [2]International Innovation Center of Tsinghua University, Shanghai, [3]Terapines Ltd, [4]Beijing Information Science and Technology University

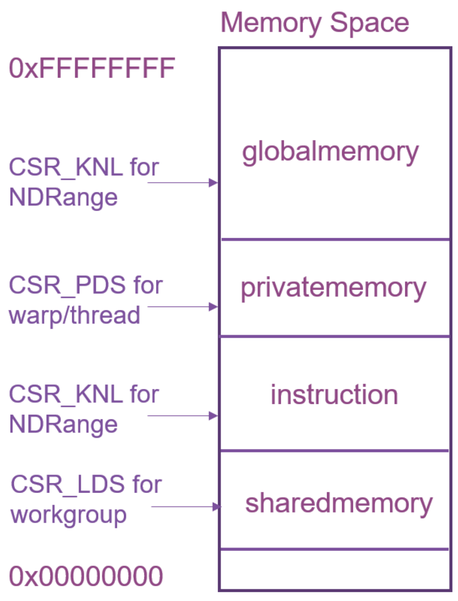
What is Ventus?

- Open-sourced RVV-based GPGPU
- An implementation of Chisel HDL, driver and compiler
- OpenCL compatibility
- RISC-V compatibility with 256 registers available



ISA: RV32IMA_ZFinx_Zicsr_V

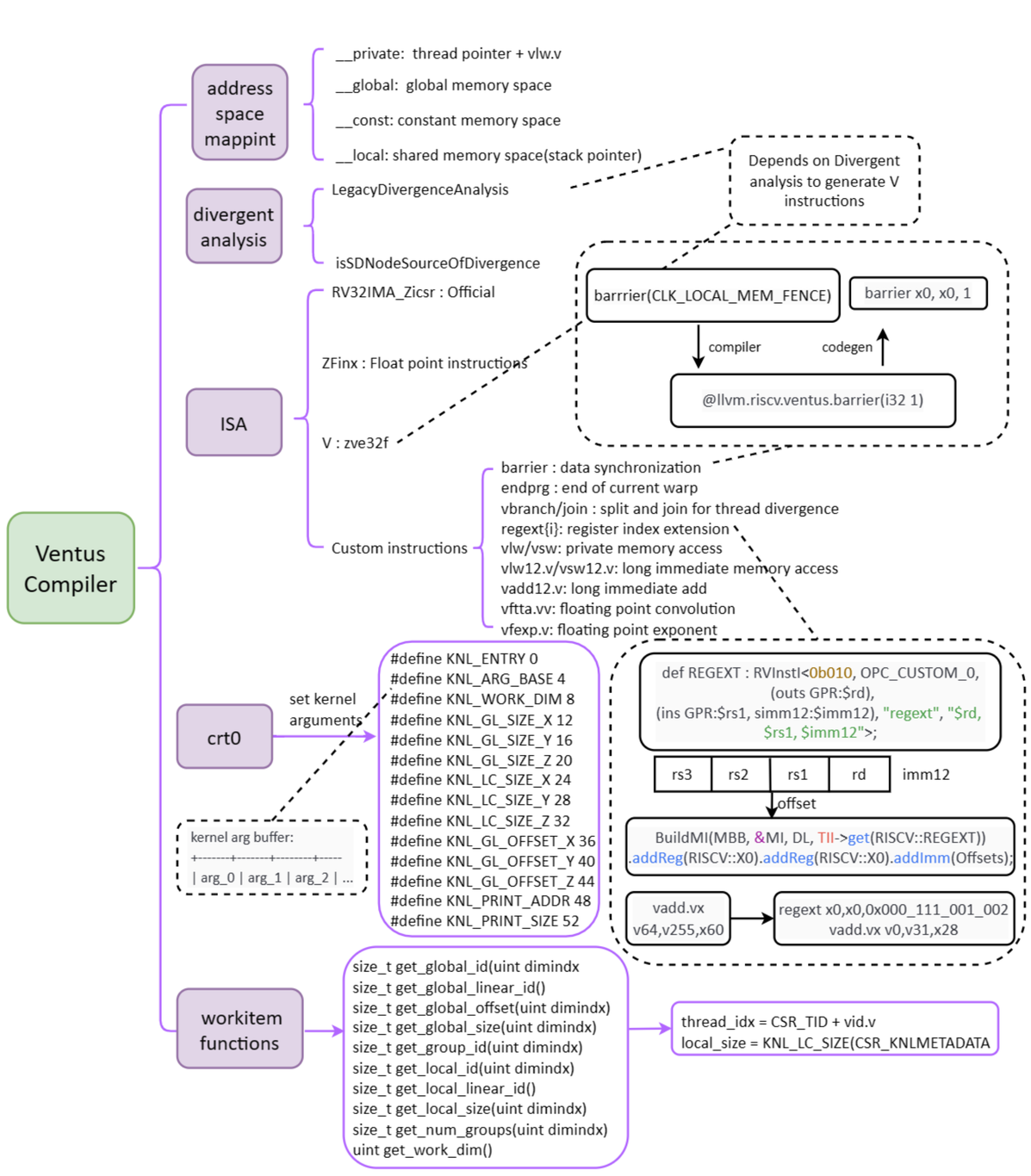
- Vector instructions for per-thread operation, elen=32 bit, vlen=32*elen
- Scalar instructions for common data
- Custom instructions:
  - VBranch/Join to control thread divergence
  - EndProgram and Barrier to control warps
  - RegisterExtension to extend register index
- Registers: 64 sGPRs, 256 vGPRs
- Memory space definition and access methods
- Custom CSRs and metadata to launch workgroup and implement workitem functions



	AMD	NVIDIA	Intel	Vortex	Ventus
ISA	RDNA	PTX	GEM	RISC-V IMF	RV32V
Instruction Length	32/64 bit	128 bit (SASS)	128 bit	32 bit	32 bit
Memory Model	GDS, LDS Constants Global	Shared, Texture Constants Global	Software Managed	Shared Global	Private Shared Global
Threading Model	workgroup wavefront 32/64 thread	CTA warp 32 thread	Root Thread Child Thread	compute unit wavefront	workgroup warp 32 thread
Register file	256 vGPRs 106 sGPRs	Scalar	128 GRFs	32 sGPRs	256 vGPRs 64 sGPRs
Thread Control	endpgm message branch thread mask	branch predicate	message branch SPF Regs split/join	thread mask (split/join)	endprg branch thread mask (vbranch/join)
Synchronization	barrier wait_cnt	barrier membar	wait fence	barrier flush	barrier fence
Execution Unit	ALU memory Matrix Core	ALU memory Tensor Core	ALU memory Matrix Engine	ALU memory	ALU memory Tensor Core

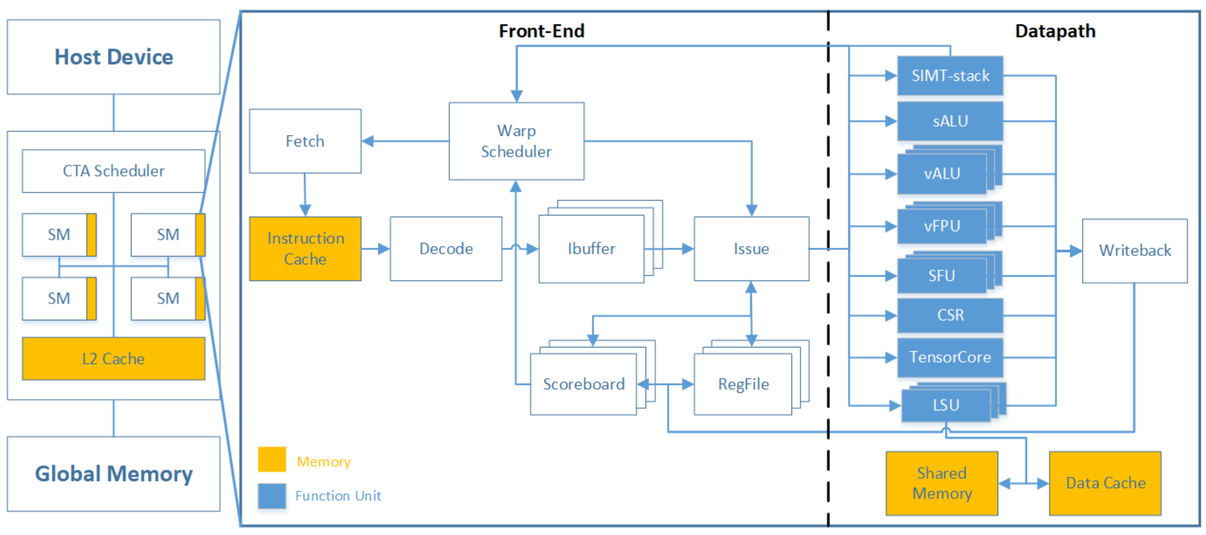
Software Stack

- Ventus-LLVM: compiler based on LLVM for Ventus ISA and library
- PoCL: OpenCL platform implementation
- Ventus-driver: KMD implementation
- Ventus-gpgpu-isa-simulator: ISS based on Spike



Microarchitecture

- Multi-level task allocation is implemented by driver and CTA-scheduler
- SM works as an RVV processor supporting warp scheduling
- 4-bank register files can be allocated according to usage
- Tensor Core supports custom tensor operations



Evaluation & Conclusion

- A complete implementation of GPGPU based on RVV
- Chisel HDL, configurable in num of warps, threads, SMs, lanes…
- A 16SM-16warp-16lane version with Tensor Core occupies 65% of the area of 4 VU19P FPGAs

- Open-sourced at https://github.com/THU-DSP-LAB/ventus-gpgpu