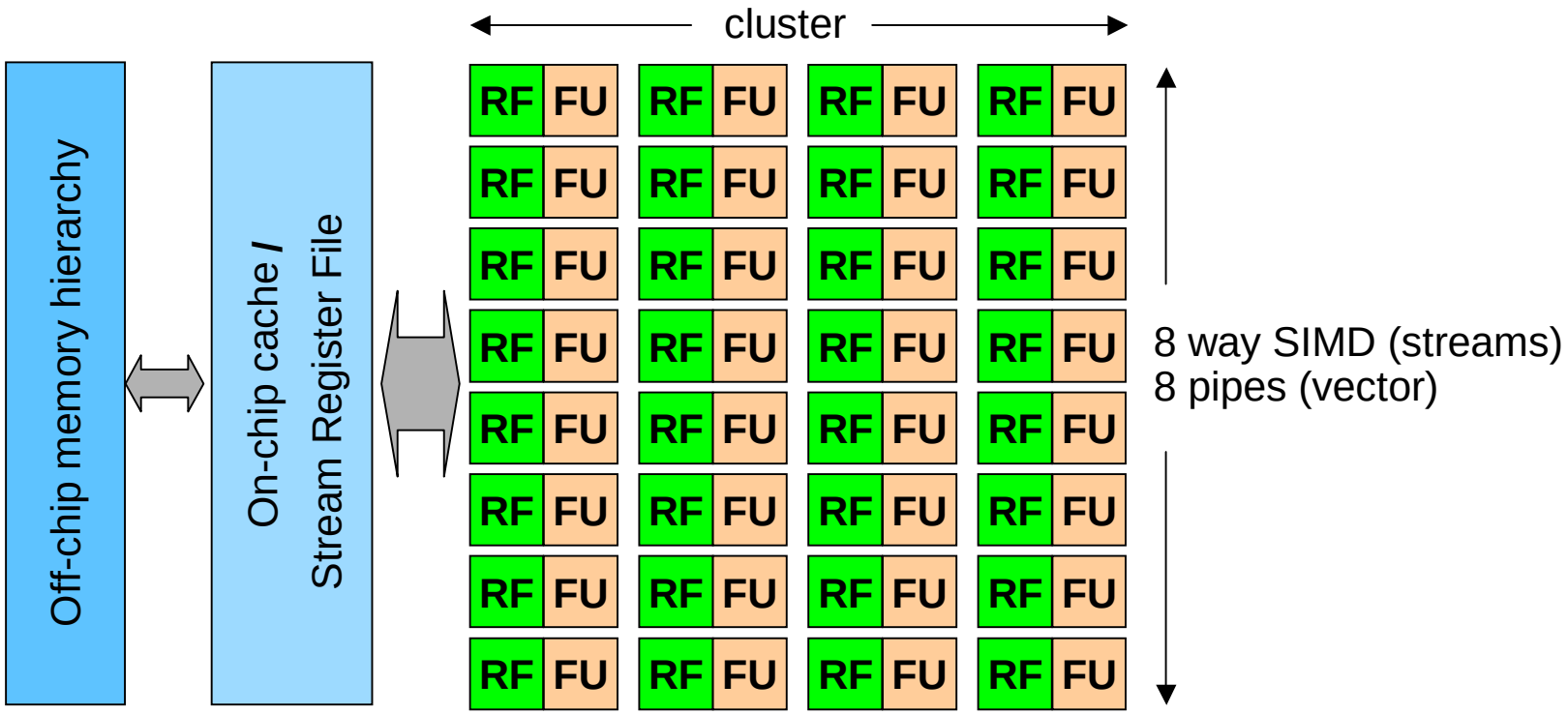
Vectors as Stream Processors

Steve Scott

Cray Inc.

sscott@cray.com

Vectors and Streams



- **Streams (SS) are like an 8-pipe vector proc with VL=8**
  - **8-way SIMD in vertical direction, scalar within FU cluster**
- **Vectors are 64- or 128-way SIMD**
  - **8-16 elements per pipe**

Similarities Between Vectors and Streams

- Mechanisms for pipelining data from/to memory
- Register file/FU clustering
  - Vector pipes (aka lanes)
  - Vector clusters within pipes
- Streaming w/ kernel locality ⇔ Outer-loop vectorization
- Stream loads ⇔ Vector prefetching &/or load buffers
- SRF accesses with different offsets ⇔ cache reuse or cross-pipe data movement

Vector Advantages

- Dynamic adaptation to unknown latencies and unknown control flow
- Can tolerate dependency chains via vectorizing

  (streaming requires aggressive loop unrolling)
- Lower instruction throughput and code size
- Existing technology with mature software

Vector Disadvantages

- When loops (VL's) are short, waste VR space
- Gathered structs a challenge
  - add stream register file / large compiler managed storage
  - or rely on spatial locality via on-chip caches
- Produce more temporaries (VL=64/128 vs. 8)
  - but is this a problem?
  - cluster vector register files can be larger than cluster scalar register files (restricted access, single rd/wr RAMs)
  - With decent sized Vreg file, don’t see register pressure, even for outer-loop vectorization