Stream Register Files with Indexed Access

Stream Register File (SRF)

- SRF serves two primary purposes in a stream processor:
  - Capture stream temporal locality
  - Staging area for memory transfers
- The SRF is implemented as N banks, each associated with a single compute cluster
- Unlike a cache, the SRF is software managed, and is a separate address space from the memory system
- Data needed by a particular cluster is placed in the SRF bank associated with it, thereby reducing communication requirements

Locality in Stream Programs

- Conventional SRF implementation only captures in-order stream accesses
- Indexed SRF allows reordered access patterns
- 2 degrees of indexing freedom:
  - In-lane: a cluster can index to local SRF bank only
    - Restricted access freedom
    - Lower hardware overhead and complexity
    - Peak bandwidth comparable to sequential streams
  - Cross-lane: any cluster can index to any location of SRF
    - Greater flexibility in access patterns
    - Higher hardware overhead and complexity
    - Lower peak bandwidth due to cross-lane communication

SRF with Indexed Access

- Indexed SRF allows reordered access patterns
- Supports more concurrent logical streams than available hardware streams
- Local register spilling to SRF
- Unified storage for streams and scratchpad memory
- High bandwidth scratchpad accesses

Future Directions

- Benchmark Results
  - FFT_2D: 64x64 2D FFT models multi-dimensional accesses
  - Rijndael: Encryption algorithm with large numbers of data-dependant table lookups
  - IG_: Parameterized irregular grid neighborhood access synthetic benchmark. Parameter values for data sets are as follows:

<table>
<thead>
<tr>
<th>Data set</th>
<th>Compute density</th>
<th>Avg. graph degree</th>
<th>Avg. strip size</th>
</tr>
</thead>
<tbody>
<tr>
<td>K0_SML</td>
<td>16</td>
<td>4</td>
<td>1812</td>
</tr>
<tr>
<td>K0_SCL</td>
<td>51</td>
<td>4</td>
<td>1812</td>
</tr>
<tr>
<td>K0_DMB</td>
<td>16</td>
<td>16</td>
<td>267</td>
</tr>
<tr>
<td>K0_DCS</td>
<td>51</td>
<td>16</td>
<td>267</td>
</tr>
</tbody>
</table>

- Measurements obtained for Imagine machine model (8 clusters, 4K SRF words/cluster, no stream cache, 2.6 GB/s peak memory system bandwidth)

Performance Impact of Indexed SRF Access

- Eliminates stream reordering memory operations
  - Reduces memory bandwidth demands
  - Shortens lifetime of data set in SRF, reducing the number of live data sets and hence increasing the strip size

- Eliminates data replication in SRF and redundant memory accesses
  - Reduces space occupied in SRF, allowing for longer strip sizes
  - Reduces memory bandwidth demands by eliminating repeated accesses

- Kernel K1 generates matrix M, K2 consumes M'
- Sequential streams (i.e. no SRF indexing) requires transpose to be done through memory
  - With indexing, M' read directly from M in SRF

- Kernel takes stream of nodes and stream of neighbors
  - W/o indexing, repeated elements are replicated in neighbor stream
  - SRF indexing allows a single copy to be referenced multiple times

SRAM Array Implementation

- Each SRF bank implemented as 4 sub-arrays due to VLSI constraints
- Leverage sub-banks to support up to 4 concurrent indexed access per bank
  - Low additional overhead:
    - Row decoder per sub array
    - 8:1 muxes at output of sub arrays
- Sequential streams perform wide accesses from a single sub array at a time
  - Energy efficiency of sequential stream accesses comparable to non-indexed SRF

Future Directions

- Leverage for current Merrimac applications
  - StreamFEM:
    - Explicit repeated accesses of neighbor cells in 3D mesh for flux calculation
  - StreamFio:
    - Data reordering for stencil accesses
    - High bandwidth scratchpad accesses
- Explore other potential application targets
  - Sparse matrix-vector multiply, irregular meshes, sorting etc.
- Methods for extracting indexed access patterns from high-level code
- Full support in compilation and simulation infrastructure