ElasticFlow

A Complexity-Effective Approach for Pipelining Irregular Loop Nests
Paper by Zhiru Zhang, Mingxing Tan, Gai Liu, Ritchie Zhao, and Steve Dai
Presented by Grace Cimaszewski, Guangwei Chen, and Shaan Shetty
Outline

- Why ElasticFlow

- Problem: unrolling irregular loop nests
  - Definition: program structure which contains one or more dynamic-bound inner loops
  - Cannot determine loop bound statically at compile time: for example, how to find the length of a linked list?
  - Using static loops for some algorithms is resource expensive (examples: Hashing & Chaining)

- ElasticFlow can allocate hardware for the common case, not worse case (traditional approach)
Irregular Loop Nests

• In keysearch for a hash table, most efficient hashes have a unique key. Conflicts and key chaining is rare.

• But we still have to consider the worst case condition, rather than just using an irregular while() loop (as in (b)), if we need to pipeline (in (c)).

• This is wasteful in terms of resources but yet modern HLS tools cannot efficiently pipeline irregular loops.
ElasticFlow Architecture

- sLPA Architecture
  - Static Scheduling
  - ElasticFlow’s strength: dynamic scheduling

- mLPA Architecture
Baseline Approach

- Currently taken by existing HLS tools
- Every inner loop iteration executes serially.
  - Low Throughput
  - Pipeline frequently stalled
  - Inner loop acts as bottleneck for the Pipeline
Single Loop Processing Array (sLPA)

- Static Scheduling
  - Stage B is loop processing array consisting of N LPUs
  - Outer Loop iteration assigned one LPU based on its modulo N value
  - Throughput is improved
  - Resource efficiency is poor as the workload is skewed towards LPU1

Solution?
Single Loop Processing Array (sLPA)

- Dynamic Scheduling
  - outer loop dispatches its dynamic-bound inner loop to a free LPU
  - Improvement in both throughput and resource efficiency
  - Out of order execution: reorder buffer required

Will this work for multiple variable bound loops?
Extending sLPA to Multiple Variable Bound Loops

- Workload unbalanced
  - idling in sLPA D as sLPA B bottlenecks pipeline progress

- Hardware resource wastage

- Problem is again static resource assignment for elastic workload
Multiple Loop Processing Array (mLPA)

- Dynamically reallocated LPUs for loop B or D depending on workload distribution.
- Reusing Hardware (improved resource utilization)
- Reduced stalling (No bottleneck due to larger iteration loops)
Example of mLPA reducing stalls and resources

- Dbjoin program: merges two hash entries with the same key
- Two separate irregular loops: stage B and D
- Using mLPA reduces number of LPUs required and also reduces stalling
ElasticFlow Synthesis

-LPU Allocation

-Distributor & Collector Synthesis

-Deadlock Avoidance

-Buffer Sizing
Problems and Constraints

- Distributor/collector coordination
- Deadlock avoidance
- Choice of ROB (reorder buffer) size
  - Consider the maximum, minimum, and average latencies and the standard deviation of the overall loop latency
- ElasticFlow assumes there are no outer loop carried dependencies involving stages synthesized to LPAs
LPU Allocation

Hardware area constraints: Area                  Performance (throughput)

sLPU (less area) \(\rightarrow\) mLPU: (better performance)

Objective: optimizes performance / resource usage (ILP)
- classify LPU into different types
- assign different types LPUs to different LPAs
- maximize resource usage

\[
\text{maximize } \sum_{k=1}^{K} \sum_{i=1}^{N} D_{k} r_{ik} + \beta \sum_{k=1}^{K} n_{k}
\]
Distributor and Collector Synthesis

Distributor → LPAs → Collector

Priority Outer Loop ID: small → large

Back pressure stop further distribution
Deadlock Avoidance

- Cannot accept new data from producer because no slot available in ROB
- Cannot free slots because lacking data needed to proceed

# of in-flight outer loop iterations < # of available entries in the ROB
Buffer Sizing

- ROB: hold data used by the LPAs pipeline
  \[ S = \frac{L_{max}}{\max(L_{avg} - 3\sigma, L_{min})} (K - 1) + 1 \]
- Delay Buffer: hold data not used by the LPAs pipeline (similar to ROB)
  > number of possible in-flight loop instances in the LPA

Distributor → LPAs → Collector → Other stages proceed down the pipeline

Delay buffer
Implementation

- ElasticFlow is implemented as an additional LLVM pass for the HLS tool and is applied after compilation and optimizations.
- Normal bounded loops are pipelined using the default algorithm in Vivado 2014.1.
- LLVM pass also inserts FIFOs between different stages and annotates dataflow directives to force different stages to run in a pipelined dataflow manner.
Experimental Results

- Speedup roughly proportional to number of LPUs used

- Comparing ElasticFlow to aggressive worst-case inner loop unrolling: similar latency, with 3-4x reduction in FF, LUT, and slice usage

- On benchmarks with multiple inner loops: mLPU can improve upon sLPU by a factor of up to .34x
Citations

Quiz Question

What are some of the characteristics of an mLPA, as compared to sLPA?

1. Smaller area, less performance
2. Larger area, less performance
3. Smaller area, greater performance
4. Larger area, greater performance