Scotch: Generating FPGA-Accelerators for Sketching at Line Rate

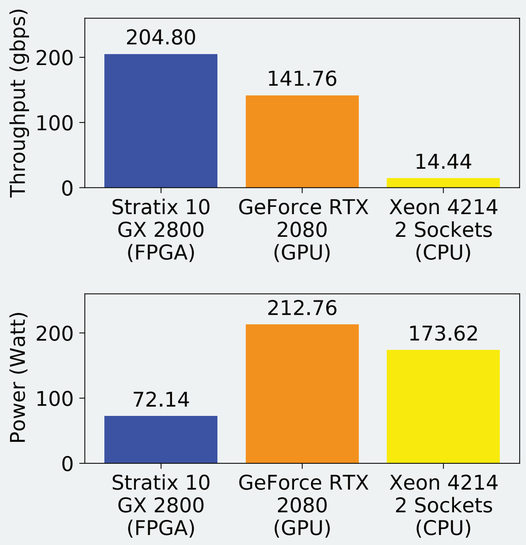
Martin Kiefer, Ilias Poulakis, Sebastian Breß, Volker Markl

Background & Motivation

Sketching
- Constructing stream summaries
  - e.g., Count-Min, AGMS, HLL, ...
- Various applications (e.g., AQP, ML, Network)

FPGAs
- Custom hardware defined by software
- Reprogrammable
- Circuit-level parallelism
  - Data / Task / Pipeline

Sketching on FPGAs
- High guaranteed throughput
- Lower power consumption
- Boards for various interconnects and use cases



Challenges & Approach

An FPGA expert is required!

Device-, vendor-, interconnect-specific implementations
- Reusing existing artifacts is hard

Register-Transfer Level (RTL) programming
- Concepts unfamiliar to software developers

Scotch's Approach: Abstract & Automate

Device & I/O agnosticism
- Sketching & I/O with common interface

Lightweight sketch specification
- Programming models + DSL

RTL generation
- Sketching RTL generated from DSL

Manual tuning
- Sketch size vs. resources & timing
- Trial and error

Automated Tuning
- Optimization in a feedback loop

Scotch: System Overview

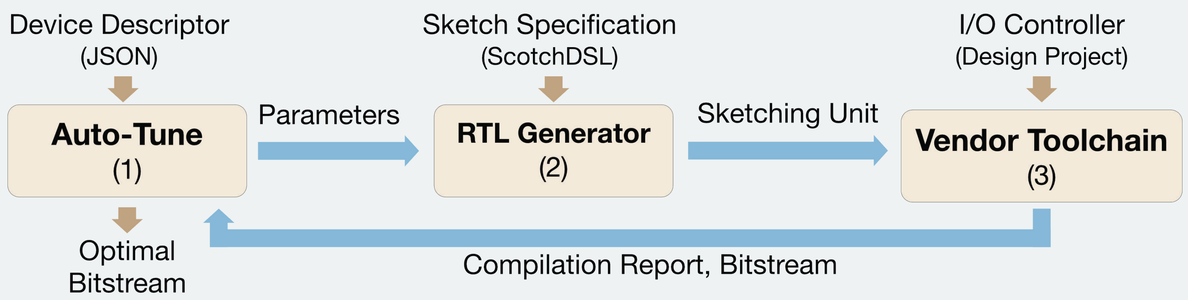
(1) Auto-Tune proposes a sketch size to the RTL generator

(2) RTL generator creates sketching unit based on parameters and specification

(3) Vendor toolchain compiles the full accelerator (sketching unit + I/O controller)

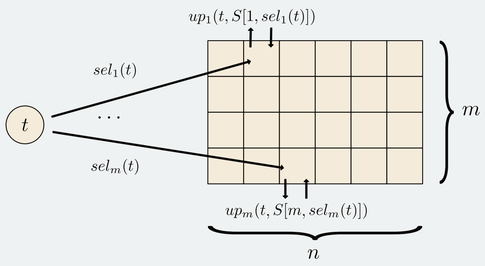
(1) Auto-Tune algorithm analyzes the compilation report and recalibrates

(...) repeat until optimal solution is found



Sketch Specification

Select-Update model for sketches
- Matrix-shaped state
  - One entry per row is updated per input value
- Select function
  - Determines one entry in each row
- Update function
  - Determines the new value of an entry



Select-Update model

ScotchDSL
- DSL for select and update functions
- Parsed by the RTL generator
- Clocks and control signals are not exposed
- Bitvectors as first class citizens
- Logic and arithmetic operations
- Restricted control flow

ScotchDSL Example: AGMS/EH3 update function

```
update(seed, v, state, outstate){
      mask <= '10101010101010101010101010101010';
      h    <= parity(v(30 downto 0) | v(31 downto 1) & mask);
      eh3 <= seed(0) ^ parity(seed(32 downto 1) & v) ^ h;

      outstate <= eh3 = '0' ? signed(state) + 1 : signed(state) - 1;
}
```

Accelerator Architecture

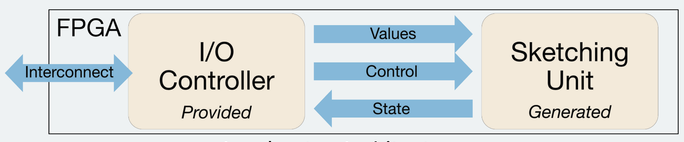
Sketching Unit
- Fully pipelined
- Data parallelism supported
- Standard VHDL

- Compute unit
  - Computes select function
  - Reads state from memory
  - Data forwarding
  - Computes update function
  - Writes state to memory

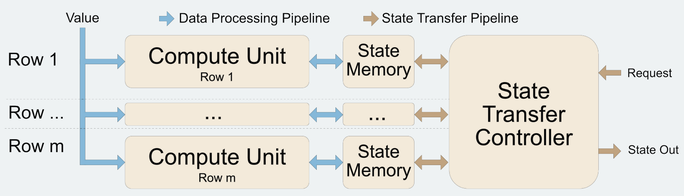
- State memory
  - Pipelined BRAM for random access
  - Register for column sketches

- State transfer controller
  - Read state from memory
  - Expose state via interface

I/O Controller
- Interfaces sketching & off-chip communication
- Vendor, board, and interconnect specific
- Provided by an expert



Accelerator Architecture



Sketching Unit Architecture

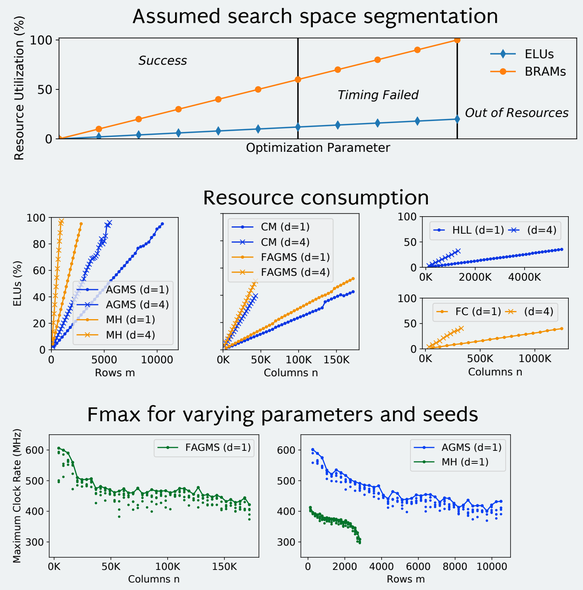
Automated Tuning

Automated tuning
- Number of rows or columns is maximized
  - subject to timing and resource constraints

Algorithm
(1) Compile initial guess
(2) Compute upper bound
- based on resource consumption
- assuming linearity

(3) Binary search for optimal parameter

Actual search space is noisy
- Randomized algorithms in vendor toolchains
- Five tries before considering timing failed



Evaluation

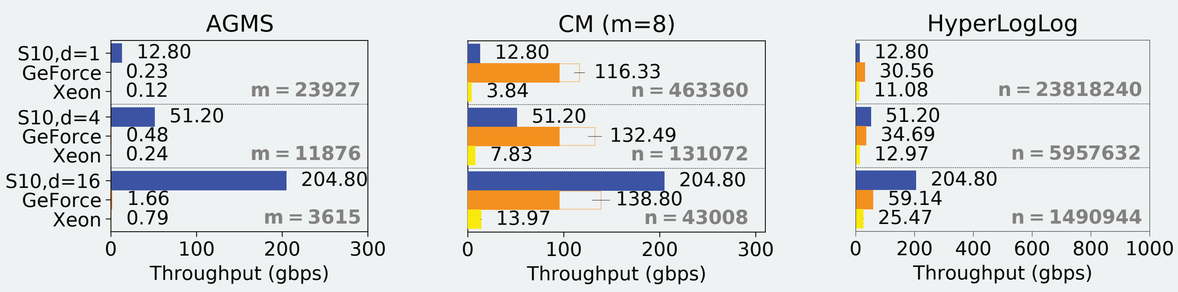
Throughput experiments
- S10: Intel Stratix 10 GX 2800 FPGA
  - Accelerator generated by Scotch
  - Data parallelism degree d 1/4/16
- GeForce: Nvidia GeForce RTX 2080 GPU
  - CUDA, Hand-tuned
- Xeon: Intel Xeon Silver 4214, 2 Sockets
  - OpenMP, Hand-tuned

- Uniform data distribution

More experiments in the paper
- Resource consumption, fmax
- Power consumption
- Xilinx devices
- Comparison to hand-tuned implementation

Generated accelerators outperform parallel GPU and CPU baselines
- Data parallelism is crucial

Trade-off between sketch size and degree of data parallelism / throughput



Column sketch | Matrix sketch | Row sketch

Conclusion

Scotch generates accelerators that ...
- provide high throughput
- with a low energy footprint

with a holistic approach that ...
- abstracts from vendors and interconnects
- generates code for sketching based on a DSL
- tunes the implementation automatically
- does not require an FPGA expert in the loop

Further interesting content in the paper
- Strategies and models for data parallelism
- Details on RTL generator and architecture
- Extensive evaluation

Author Affiliations

Contact:
martin.kiefer@tu-berlin.de

Source & Data on GitHub:
martinkiefer/scotch

Funding

Bundesministerium für Bildung und Forschung

This work has received funding by the German Ministry for Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (01IS18025A, 01IS18037A) and Software Campus (01IS17052).