Automata Processing on FPGAs

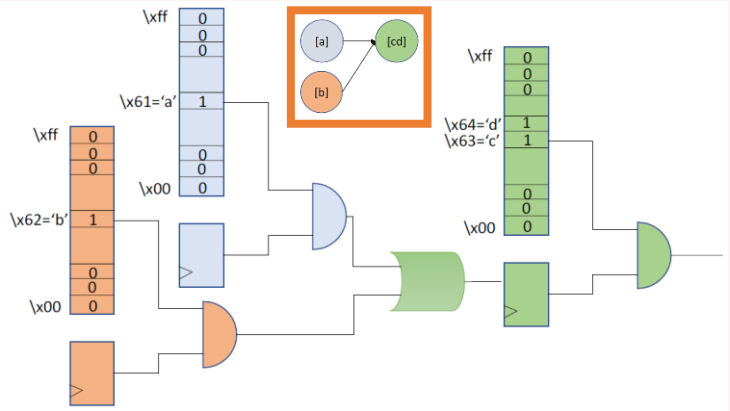
Chunkun Bo, Vinh Dang, Ted Xie, Jack Wadden, Mircea Stan, Kevin Skadron
Department of Computer Science, University of Virginia, chunkun@virginia.edu

Motivation

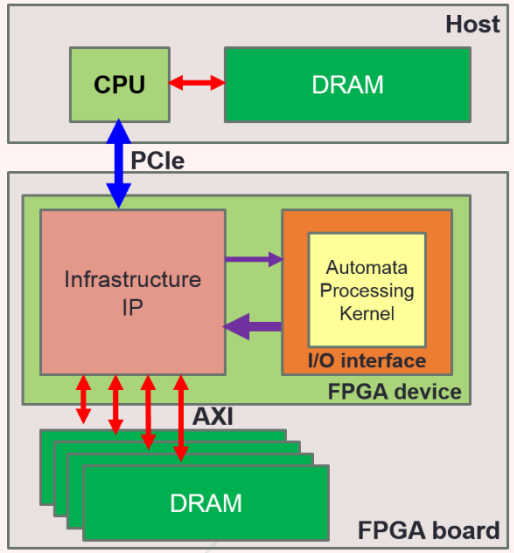
- Automata processing has shown its capability in a variety of applications: network inspection, machine learning, bioinformatics, data mining, etc.
- Automata processing is challenging on von Neumann architectures, esp. CPUs, for large, complex automata
- Spatial architectures: process many automata transitions in parallel by laying out the automata directly on hardware
  - Automata Processor (AP): an ASIC specifically designed for parallel automata processing by Micron; each board is equipped with 32 chips and each chip has 49,152 states
  - FPGAs: massive hardware resources such as registers and lookup tables, over 95,000 states on Virtex VU9P

REAPR Design

- REAPR: Reconfigurable Engine for Automata Processing on FPGAs
- Each STE is one-hot encoded as a 1x256 memory column and is AND'ed with an activation state register



REAPR Design



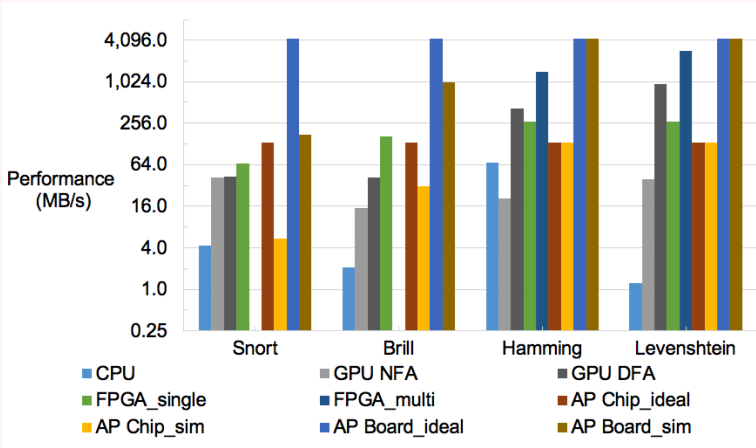
- The first work that includes I/O circuitry and provides an end-to-end automata processing engine on FPGAs
- Adopt a high-level synthesis-centric approach
  - Design a dummy kernel with I/O interface using Vivado HLS
  - Replace the dummy kernel with the actual automata processing kernel and compile the newly integrated kernel
- Open source: https://github.com/ted-xie/REAPR

New Features

- Automated integration
- Deploy new workflow on AWS F1 and Nimbix
- Processing multiple symbols to achieve higher throughput
- Fast, symbol-only reconfiguration for large datasets
- Simpler integration phase
- Faster reporting architecture

Evaluations across Platforms

- Four different Platforms: CPU, GPU, Automata Processor, and FPGA
- Various applications from ANMLZoo benchmark suite (only four shown below)



- REAPR works much faster than CPU and GPU methods
- REAPR is competitive with AP

Conclusions

- The first end-to-end automata processing engine on FPGA
- Provide a whole workflow in cloud (AWS EC2 F1)
- Achieve promising results against existing methods

Acknowledgements

This work was supported in part by NSF grant no. CCF-1629450, a grant from Xilinx, and support from CRISP, one of six centers of JUMP, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.