Rosetta: A Realistic Benchmark Suite for Software Programmable FPGAs

Udit Gupta, Steve Dai, Zhiru Zhang

Computer Systems Laboratory, Electrical and Computer Engineering, Cornell University, Ithaca, NY

Overview

Realistic benchmarks are important for the development of sophisticated yet scalable **high-level synthesis (HLS)** tools that allow FPGAs to be programmed in software while achieving high-performance designs.

- We propose a suite of realistic software applications with enforceable **system-level hardware constraints** to model hardware accelerators targeting heterogeneous FPGAs.
- We present a C/C++ and OpenCL design and verification flow that accommodates both the sequential and parallel programming models commonly supported by HLS.

The Three R's of Rosetta

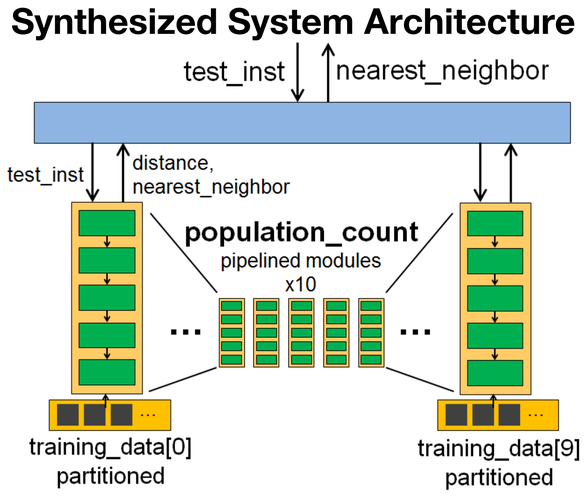
- **Realistic**: Representative application domains with large applications and user-enforceable design constraints.
- **Reducible**: Users can profile applications and sub-kernels to productively benchmark QoR.
- **Retargetable**: Intended to be vendor-agnostic and target various FPGA devices.

K-Nearest Neighbor Digit Recognition

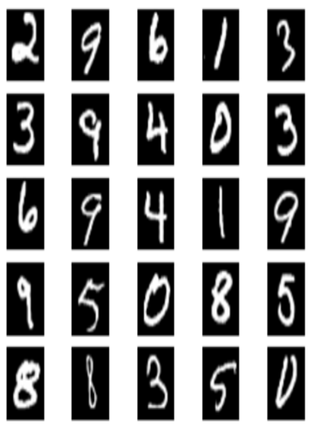
Source Code

```
for digit in range (0, 10):
  for training_inst in training_data[digit]:
  diff = training_inst xor test_inst
  distance = population_count(diff)
  if (distance < min):
    nearest_neighbor = training_inst
    min = distance
return nearest_neighbor
```

Synthesized System Architecture



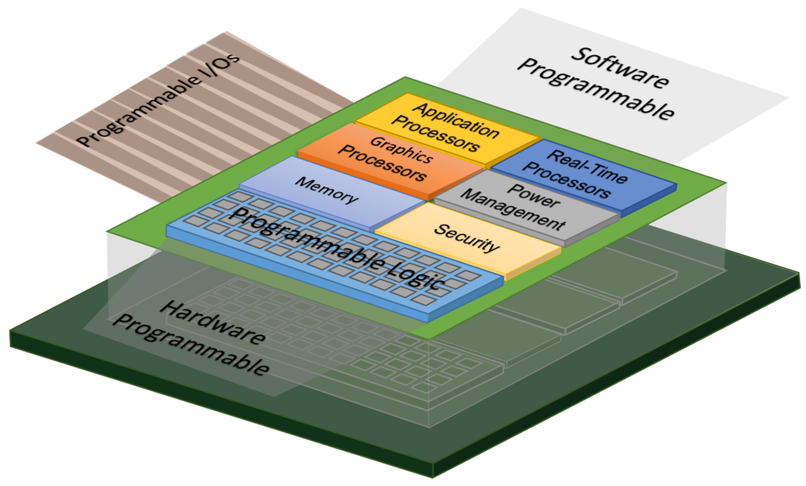
Source Data

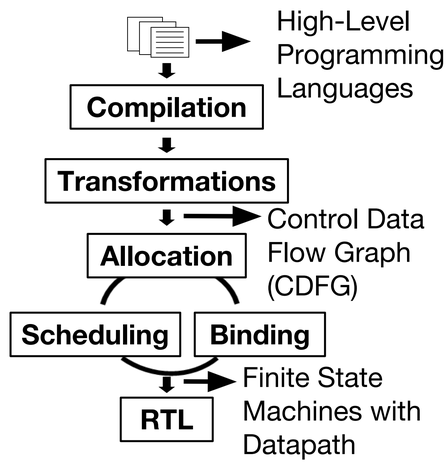


- Efficient hardware for population count kernel.
- Parallel execution (by **loop unrolling**) across digits 0-9.
- **Pipelined** execution over training instances.
- **Partitioned** training data increases memory bandwidth.

2-3x runtime speedup observed using FPGAs

Modern FPGAs and HLS Tools



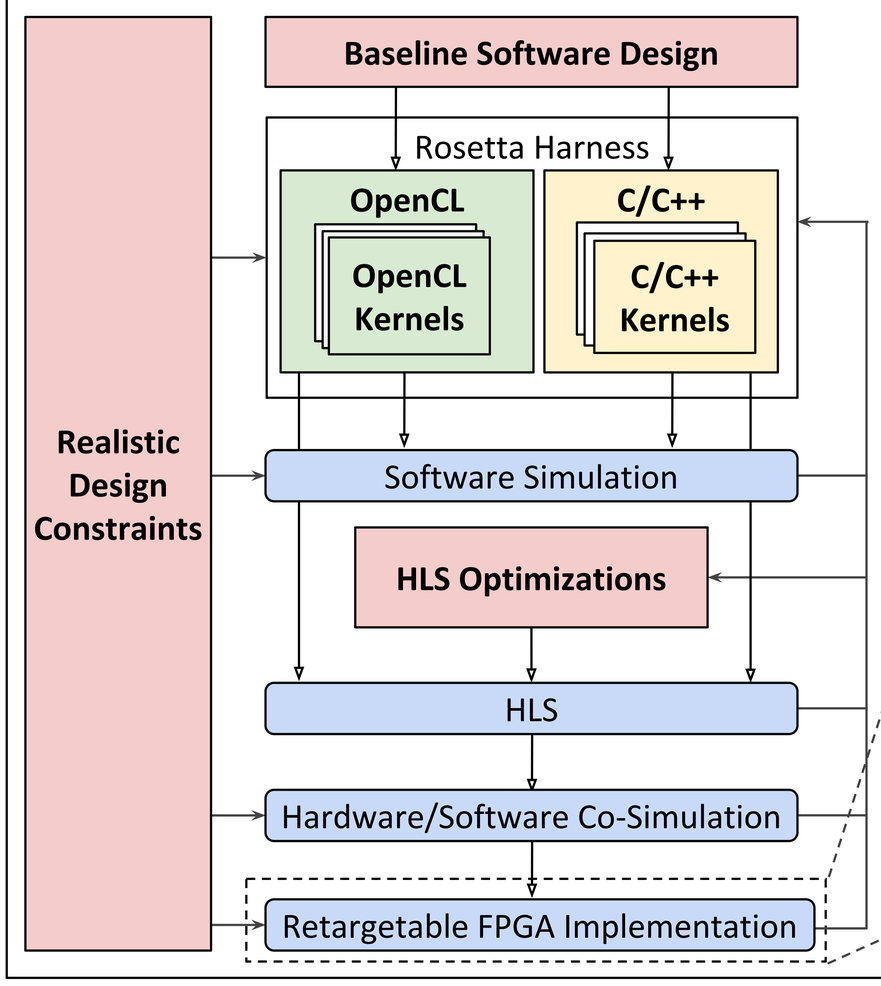


- Modern FPGAs are **heterogeneous SoCs** composed of multi-core processors, hardened IPs, and reconfigurable fabric.
- Calls for an unprecedented amount of **software programmability** as FPGA emerges from a logic to computing device.
- State-of-the-arts HLS tools perform scheduling and binding to convert untimed behavioral design in software programming languages to timed cycle-accurate RTL optimized for performance, area, and power.

Existing HLS Benchmark Suites

- Span a limited subset of computing domains and consist of **small kernels** (less than 1000 lines) which lack the complexity representative of common HLS designs.
- Provide no support for **multiple programming models** and require significant user overhead in adding HLS-specific **optimization directives**.
- Do not allow convenient **parameterization** of designs by user.

Rosetta Development Flow and Application Architecture



Development Flow

- C-based application-specific kernels can be easily added to the generic harnesses.
- Synthesis results are used to optimize designs given design constraints.

FPGA Implementation

- **Host Application** - Responsible for controlling the system and the accelerator.
- **Kernels** - Consist of the compute-intensive portions of the applications to be accelerated.

Proposed Initial Benchmarks for Rosetta

Applications	Domains	Constraints	Kernels/Algorithms
Voice Removal and Pitch Shifting	Audio Processing	Latency (Real-Time)	FFT, Inverse FFT, DSP Filters
DNA and Protein Sequencing	Bioinformatics	Throughput	Smith Waterman
Advanced Encryption Standard	Cryptography	Throughput	Matrix Substitutions and Permutations
Monte Carlo Option-Pricing	Financial Analysis	Throughput / Latency	Black Scholes, Mersenne Twister, Box Muller
Digit Recognition	Machine Learning	Throughput	Population-Count, K-NN, K-Means
Convolutional Neural Networks	Machine Learning	Throughput	Convolution, Soft Max and Max Pool Layers
Face Detection	Video Processing	Throughput	Viola Jones Algorithm
Lane Detection	Video Processing	Latency (Real-Time)	Edge Detection