How Many CPU Cores is an FPGA Worth?

Authors: Mikhail Asiatici, Damian Maiorano, Paolo Ienne

Karthik B K <karthik.bk@incoresemi.com>

07 February 2023

Outline

1. An Overview
2. The Study
3. The Big Question

An Overview

The Motivation

- FPGAs - best performance per watt?
- Design complexity vs designer productivity
- Short turn-around times (back to Moore's law?)

This Paper

This paper -

- Presents a hybrid version of pS^5[1]
- Evaluates the performance on CPU-FPGA hybrid systems
- Presents the reality of CPU-FPGA hybrid systems as it is today

[1] state-of-the-art string sorting algorithm for multi-core shared memory CPUs

The Potential

- Irregular memory access patterns – need solving? (described in a later paper)
- Average RTL designer can be more productive
- Are domain-specific accelerators the way to go? (open question)

The Study

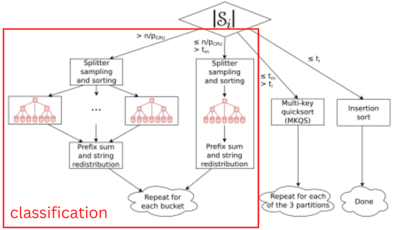
A Quick Look At S^5, pS^5, and pHS^5

- Quick sort, sample sort – the basis
  - pick a pivot, move it to the end, iterate till $l < r$
  - pick ’p’ pivots – voila! parallelization
  - S^5 – super alphabets - faster comparisons[2]
- pS^5 – the kernels
  - Classification - based on common prefixes and splitters
  - MKQS and insertion sort – size based

[2] LCP grows in size every iteration

The Classification Step

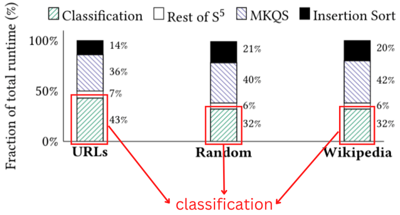
- massively parallel
- sorts into buckets based on $<, =, >$
- each splitter's tree – one PE in the AFU



$|\mathcal{S}_i|$
$> n/p_{CPU}$
$\leq n/p_{CPU}$
$> t_m$
$\leq t_m$
$> t_i$
$\leq t_i$
Splitter sampling and sorting
...
Prefix sum and string redistribution
Splitter sampling and sorting
Prefix sum and string redistribution
Repeat for each bucket
Multi-key quicksort (MKQS)
Insertion sort
Repeat for each of the 3 partitions
Done
classification

The Runtime

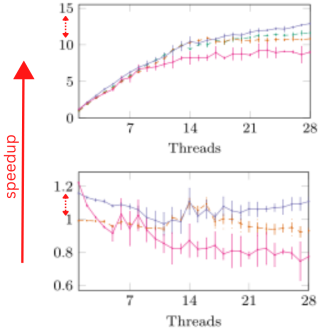
- Classification – massively parallel – easily split to PEs
- MKQS – includes classification [a]



[a] generalization of the three- way partitioning step

The Speedup

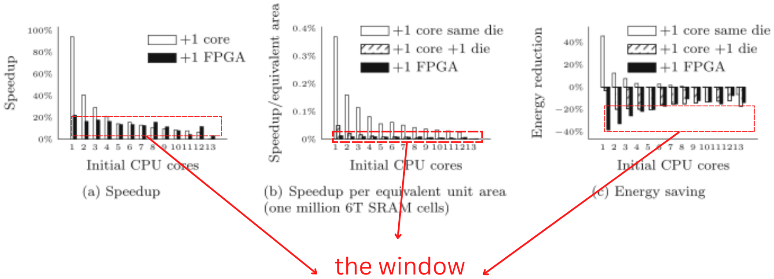
- adding 1 FPGA – 6 PEs (full infra)
- data is english text (complete sentences)
- longer LCPs are more frequent
- (upper) with single CPU thread
- (lower) with multiple (nproc) CPU threads
- **notice Y scale



adding 1 FPGA -- 6 PEs

The Big Question

What Did The Authors Learn?



(a) Speedup

(b) Speedup per equivalent unit area (one million 6T SRAM cells)

(c) Energy saving

the window

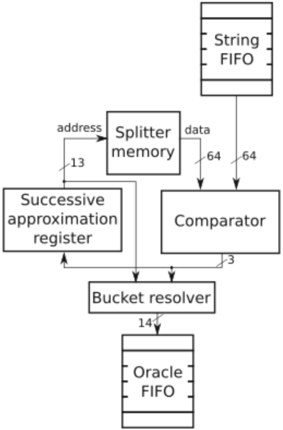
What Did The Authors Learn? (contd.)

- adding a cpu core on the same die – best solution
- fpga speedup is never enough – additional power consumption
- But
  - FPGA needs seperate infra
  - changing the number of dies – adding cpu/fpga
  - pS^5 specific – parallelizable runtime   100%
  - better ocm – more speedup?
  - will this ever be feasible? (open question)

Open for Discussions

FPGAs are not yet a clear winner when competing with complex highly optimized parallel software running on high-performance CPUs.

Inside a PE


String FIFO
address
Splitter memory
data
13
64
64
Successive approximation register
Comparator
3
Bucket resolver
14
Oracle FIFO

Some Parameters

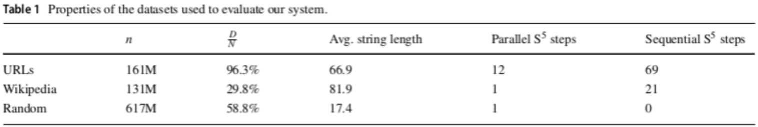


Table 1 Properties of the datasets used to evaluate our system.

	n	$\frac{D}{N}$	Avg. string length	Parallel S^5 steps	Sequential S^5 steps
URLs	161M	96.3%	66.9	12	69
Wikipedia	131M	29.8%	81.9	1	21
Random	617M	58.8%	17.4	1	0

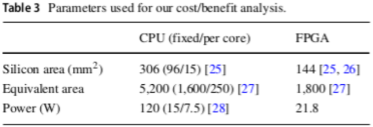
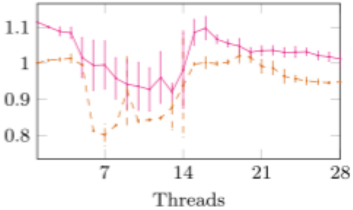


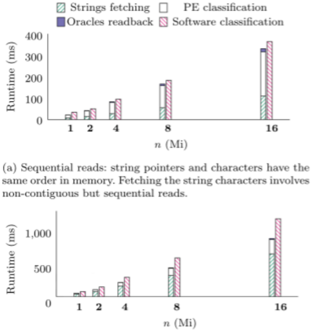
Table 3 Parameters used for our cost/benefit analysis.

	CPU (fixed/per core)	FPGA
Silicon area (mm^2)	306 (96/15) [25]	144 [25, 26]
Equivalent area	5,200 (1,600/250) [27]	1,800 [27]
Power (W)	120 (15/7.5) [28]	21.8

Some More Results



(f) Speedup on **Random** to pS^5 with the **same number of threads**



(a) Sequential reads: string pointers and characters have the same order in memory. Fetching the string characters involves non-contiguous but sequential reads.

(b) Random reads: string pointers and characters do not have the same order. Fetching the string characters involve random reads.