Performance Modeling and Analysis of Exposed Datapath Architectures

Klaus Schneider, Demyana Selim, and Nadine Kercher

Department of Computer Science
RPTU University Kaiserslautern-Landau, Germany

FDL, September 9-12, 2025, St. Goar, Germany

Motivation

- RISC architectures are limited in using instruction-level parallelism
- exposed datapath architectures expose architecture details to the compiler
- hence, the compiler can allocate the processing units and can schedule the transfer of intermediate results by the generated program
- performance models are used to determine good parameters for processor design
- performance models guide simulations by simulators and protoypes
- **we present a performance model for exposed datapath architectures**

Outline

1. Buffered Exposed Datapath (BED) Architectures
2. Dataflow Graphs and Move Code Programs
3. A Performance Model for BED Machines
4. Benchmarking the Performance Model
5. Summary

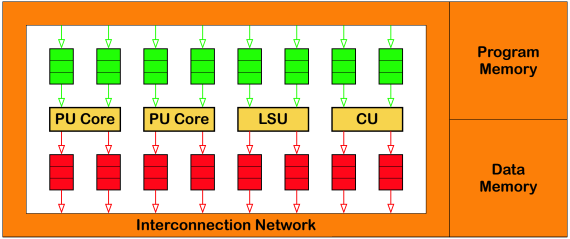
RISC Architectures

- RISC architectures dominate the processor world
- pipelined and superscalar implementations deliver high processor performance
- **however, there are also problems with RISC architectures**
  - memory access became a bottleneck
  - registers were introduced as fast on-chip memory
  - compilers focus on the effective use of registers as scarce resource
  - **number of registers limits the amount of useable ILP**
  - number of registers cannot be easily increased
- the circuit complexity also grows with $O(w^2)$ when issuing w instructions per cycle in superscalar implementations $\looparrowright$ bad power consumption

Performance Models for RISC Architectures

- instruction set simulators can be used to determine design parameters
- **in addition, several performance models exist for RISC architectures**
  - analytical performance models [14, 15, 9, 2, 12, 21, 10, 19, 1, 13]
  - empirical performance models [17, 16, 4, 3, 20]
  - sampled program simulation [18, 5, 6]
  - trend models [11, 8, 7]
- **many models focus on the size of the instruction issue width [2, 21, 12]**

Buffered Exposed Datapath (BED) Architectures



- processor is a set of interconnected processing units (PUs)
- compiler takes care of instruction scheduling, PU allocation and data transports
- IO ports of PUs have FIFO buffers to avoid synchronization of PUs

Move Code Programs and Virtual Channels

- intermediate results of PUs in BED architectures can be moved from PUs to PUs

↬ **programs of BED architectures consist of move instructions** src → tgt

- src and tgt uniquely identify I/O buffers of the PUs, constants, and opcodes
- standard PUs have two input buffers inL, inR and two output buffers outL, outR, and a further input buffer for the opcode
- move instructions like PU[i].outL → PU[i].inR transfer the head value of PU[i].outL to the tail of PU[i].inR
- **move instructions are issued by the control unit, synchronously registered at the PUs, and executed later when operands become available**

↬ dataflow style execution of a sequential program

Dataflow Graphs as Intermediate Compiler Representations

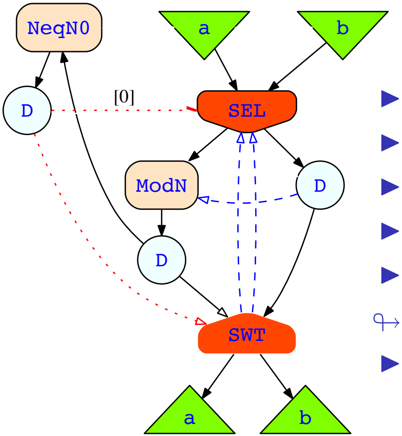
- BED architectures may execute many instructions in parallel
- to expose instruction-level parallelism (ILP) of sequential programs, we suggest the following work flow:

sequential program → dataflow graph → move code program

- nodes of the dataflow graphs can fire when operands are available

↬ dataflow graphs expose the entire ILP of the sequential program

More about Dataflow Graphs



- a fixed set of process nodes
- with static point-to-point connections
- each connection is a (unbounded) FIFO buffer
- nodes can fire if input values arrive
- values are buffered if not immediately consumed
- ↬ highly parallel MoC
- we do not need further details for the following

Performance of BED Machines

- prototypes were implemented for some BED architectures like TRIPS and others
- while reasonable design parameters might have been chosen, no proofs or arguments about their optimality were given
- simulators are required to identify relevant design parameters and their values
- **however, many simulations are required which is time-consuming**
- **performance models are required to reduce the number of simulation runs**
- in the following, we present our performance model for BED architectures

Parameters of our Performance Model

- **program parameters**
  - n move instructions
  - $\mu \cdot n$ nodes and $(1 - \mu) \cdot n$ edges in the dataflow graph
  - α nodes can be fired in parallel in average (width of the dataflow graph)
- **BED architecture parameters**
  - p general purpose processing units
  - λ cycles for average latency of a node firing
  - β entries in each input/output FIFO buffer
  - w move instructions are issued in each step by the control unit

A Performance Model for BED Machines

- Proposition: The runtime t of a program with n instructions, $\mu \cdot n$ node firings, and average ILP α on a BED machine with p PUs, buffer size β, instruction latency λ, and instruction issue width w is determined as:

$$t(p, w, \beta) = \frac{n}{\min\{\alpha, \frac{p}{\mu \cdot \lambda}, w, 2p\beta\}}$$

- BED machines should therefore be designed such that the following hold:

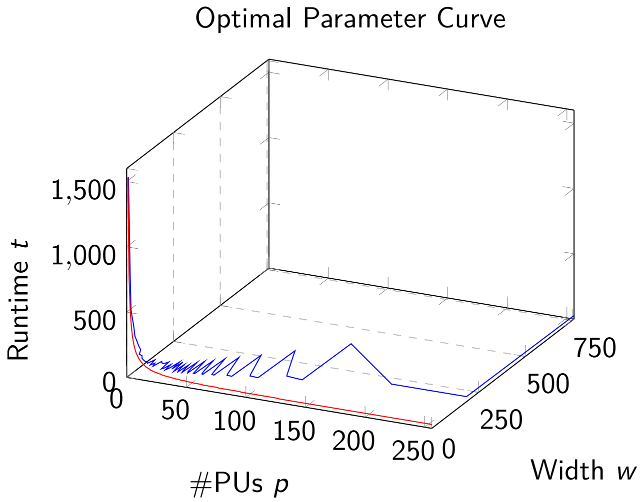
$$\frac{p}{\mu \cdot \lambda} = w = 2 \cdot \beta \cdot p$$

  - w instructions can be stored in $2p\beta$ buffer entries
  - w instructions contain $\mu \cdot w$ nodes of the dataflow graph
  - $\mu \cdot w$ nodes can be 'fired' by the p PUs with latency λ

Benchmarking the Performance Model

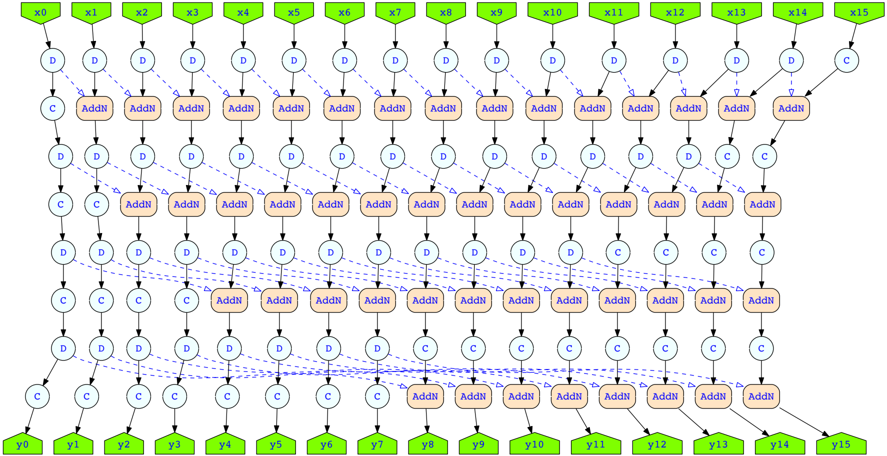
- **the performance model has been evaluated by benchmarks**
  - tree summation (of 512 numbers)
  - parallel prefix computation (of 128 numbers with Kogge-Stone and Brent-Kung)
  - odd-even transposition sort (of 16 numbers)
- **for each benchmark**
  - we first determine for a given number of p PUs, the minimal runtime t
  - for this minimal runtime t and p PUs, we minimize the instruction issue width w

Tree Summation of 512 Numbers: Optimal Parameters

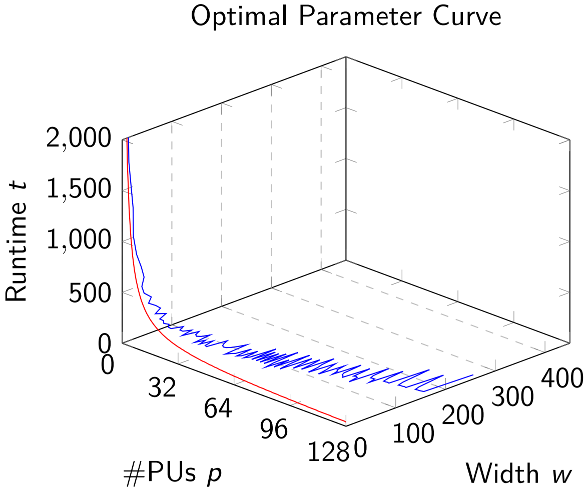


- 512 numbers were added by first adding 256 pairs, then 128 pairs, etc. in a binary tree schedule
- w grows proportionally with p as predicted by our model
- t grows anti-proportionally with p as predicted by our model

Parallel Prefix Computation by Kogge-Stone: Dataflow Graph

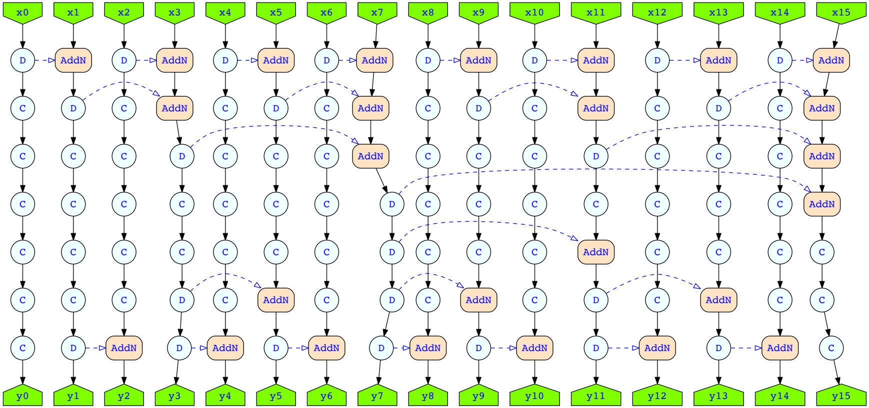


Parallel Prefix Computation by Kogge-Stone: Optimal Parameters

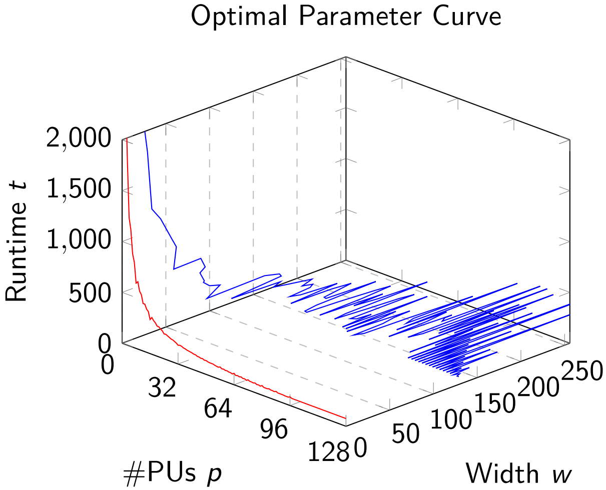


- the prefix sums of 128 numbers were computed
- t grows anti-proportionally with p as predicted by our model
- w has significant 'noise', but is enveloped by a line that grows proportionally with p
- the 'noise' is caused by mobility windows whose size depends on the remainder left by dividing the number of nodes by p

Parallel Prefix Computation by Brent-Kung: Dataflow Graph

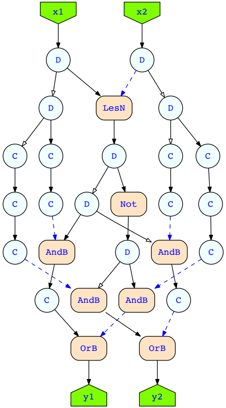


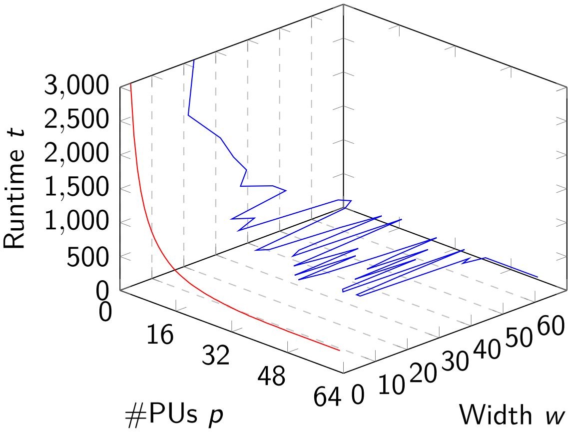
Parallel Prefix Computation by Brent-Kung: Optimal Parameters



- the prefix sums of 128 numbers were computed
- t grows anti-proportionally with p as predicted by our model
- w has significant 'noise', but is enveloped by a line that grows proportionally with p
- the 'noise' is caused by mobility windows whose size depends on the remainder left by dividing the number of nodes by p

Odd-Even Transposition Sort





Summary of Contributions

- we present a performance model for BED architectures from which we derive that balanced BED architectures require

$$\frac{p}{\mu \cdot \lambda} = w = 2 \cdot \beta \cdot p$$

- this avoids potential bottlenecks, in particular
  - ↬ every cycle, $w = 2 \cdot p \cdot \beta$ instructions can be stored in the buffers
  - ↬ every cycle, $p = w \cdot \mu \cdot \lambda$ instructions can be executed in parallel
- benchmarks proved the performance model

Conclusions

- instruction issue width w must grow proportionally with the number of PUs
- for highly parallel programs, the size of FIFO buffers is less important
- optimal parameters are not reasonable due to the anti-proportional growth since a comparable runtime can often be achieved with much smaller issue width and number of PUs

References I

[1] Breughe, M., Eyerman, S., and Eeckhout, L.
A mechanistic performance model for superscalar in-order processors.
In *International Symposium on Performance Analysis of Systems and Software (ISPASS)* (New Brunswick, NJ, USA, 2012), IEEE Computer Society, pp. 14–24.

[2] Dubey, P., Adams, G., and Flynn, M.
Instruction window size trade-offs and characterization of program parallelism.
IEEE Transactions on Computers 43, 4 (April 1994), 431–442.

[3] Eeckhout, L., Bell, R., Stougie, B., De Bosschere, K., and Kurian John, L.
Control flow modeling in statistical simulation for accurate and efficient processor design studies.
In *International Symposium on Computer Architecture (ISCA)* (Munich, Germany, 2004), IEEE Computer Society, pp. 350–363.

[4] Eeckhout, L., Nussbaum, S., Smith, J., and de Bosschere, K.
Statistical simulation: adding efficiency to the computer designer's toolbox.
IEEE Micro 23, 5 (2003), 26–38.

[5] Hamerly, G., Perelman, E., Lau, J., and Calder, B.
SimPoint 3.0: Faster and more flexible program phase analysis.
Journal of Instruction-Level Parallelism 7 (2005), 1–28.

[6] Hamerly, G., Perelman, E., Lau, J., Calder, B., and Sherwood, T.
Using machine learning to guide architecture simulation.
Journal of Machine Learning Research 7, 12 (2006), 343–378.

[7] Ipek, E., McKee, S., de Supinski, B., and Caruana, R.
Efficiently exploring architectural design spaces via predictive modeling.
ACM SIGOPS Operating Systems Review 40, 5 (2006), 195–206.

References II

[8] Joseph, P., Vaswani, K., and Thazhuthaveetil, M.
A predictive performance model for superscalar processors.
In *Microarchitecture (MICRO)* (Orlando, Florida, USA, 2006), IEEE Computer Society, pp. 161–170.

[9] Jouppi, N.
The nonuniform distriubtion of instruction-level and machine parallelism and its effect on performance.
IEEE Transactions on Computers 38, 12 (December 1989), 1645–1658.

[10] Karkhanis, T., and Smith, J.
A first-order superscalar processor model.
In *International Symposium on Computer Architecture (ISCA)* (Munich, Germany, 2004), IEEE Computer Society, pp. 338–349.

[11] Lee, B., and Brooks, D.
Accurate and efficient regression modeling for microarchitectural performance and power prediction.
In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (San Jose, California, USA, 2006), J. Shen and M. Martonosi, Eds., ACM, pp. 185–194.

[12] Michaud, P., Seznec, A., and Jourdan, S.
Exploring instruction fetch bandwidth requirement in wide issue superscalar processors.
In *Parallel Architectures and Compilation Techniques (PACT)* (Newport Beach, California, USA, 1999), IEEE Computer Society, pp. 2–10.

[13] Mitra, R., Joshi, B., Ravindran, A., Mukherjee, A., and Adams, R.
Performance modeling of shared memory multiple issue multicore machines.
In *International Conference on Parallel Processing Workshops (ICPP-WS)* (Pittsburgh, Pennsylvania, USA, 2012), IEEE Computer Society, pp. 464–473.

[14] Noonburg, D., and Shen, J.
Theoretical modeling of superscalar processor performance.
In *Microarchitecture (MICRO)* (San Jose, California, USA, 1994), IEEE Computer Society, pp. 52–62.

References III

[15] Noonburg, D., and Shen, J.
A framework for statistical modeling of superscalar processor performance.
In *International Symposium on High-Performance Computer Architecture (HPCA)* (San Antonio, TX, USA, 1997), IEEE Computer Society, pp. 298–309.

[16] Nussbaum, S., and Smith, J.
Modeling superscalar processors via statistical simulation.
In *Parallel Architectures and Compilation Techniques (PACT)* (Barcelona, Catalunya, Spain, 2001), IEEE Computer Society, pp. 15–24.

[17] Oskin, M., Chong, F., and Farrens, M.
HLS: combining statistical and symbolic simulation to guide microprocessor designs.
In *International Symposium on Computer Architecture (ISCA)* (Vancouver, British Columbia, Canada, 2000), ACM, pp. 71–82.

[18] Sherwood, T., Perelman, E., Hamerly, G., and Calder, B.
Automatically characterizing large scale program behavior.
In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (San Jose, California, USA, 2002), K. Gharachorloo, Ed., ACM, pp. 45–57.

[19] Taha, T., and Wills, S.
An instruction throughput model of superscalar processors.
IEEE Transactions on Computers 57, 3 (2008), 389–403.

[20] Wenisch, T., Wunderlich, R., Ferdman, M., Ailamaki, A., Falsafi, B., and Hoe, J.
SimFlex: Statistical sampling of computer system simulation.
IEEE Micro 26, 4 (2006), 18–31.

[21] Zyuban, V., Brooks, D., Srinivasan, V., Gschwind, M., Bose, P., and Strenski, P.
Integrated analysis of power and performance for pipelined microprocessors.
IEEE Transactions on Computers 53, 8 (August 2004), 1004–1016.