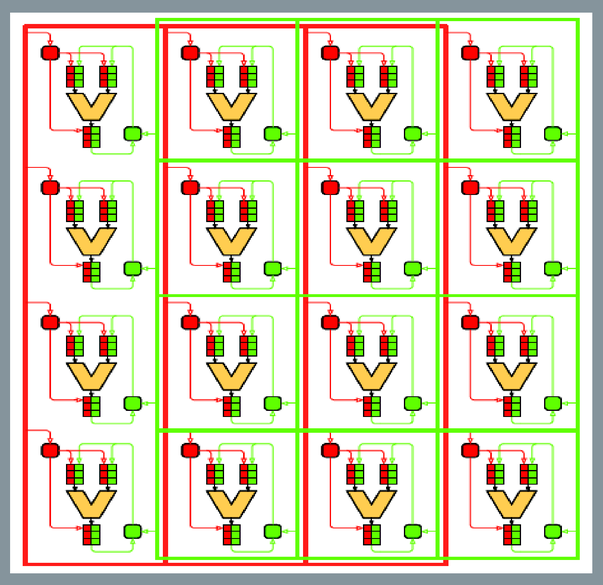
COMPUTER SCIENCE

*S*ynchronous *C*ontrol *A*synchronous *D*ataflow

SCAD : Architecture & Compilation

Motivation

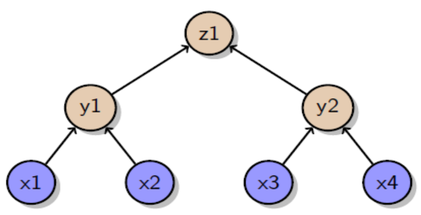
- **Instruction-Level Parallelism (ILP)**
  - restricted in VLIW and superscalar machines due to limited number of registers
    - instruction format encoding
    - complexity in register file wiring
- **Time-Predictability**
  - difficult in conventional architectures
    - cache behavior / memory access
    - out-of-order execution



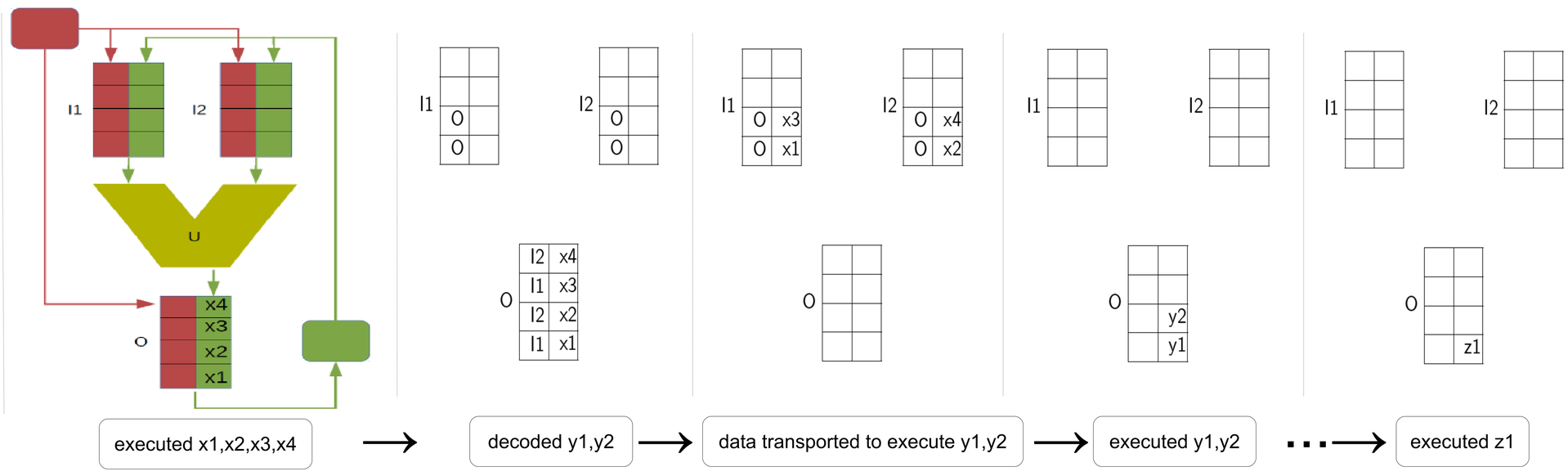
SCAD Architecture

- grid of processing units (PUs)
- FIFO buffers (queues) at every PU input and output port
- single instruction machine: move instructions *outBuf* → *inBuf*
- synchronous registration of move instructions at producer/consumer
- execution in data flow order in PUs
- asynchronous data transport between PUs
- **Application-Specific**
  - any arbitrary functionality in PUs
  - choice of different interconnection networks

Functionality

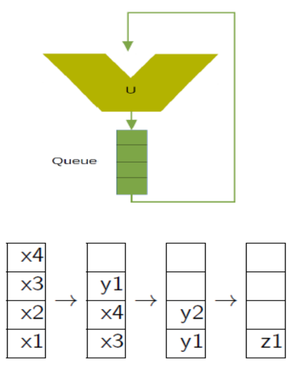


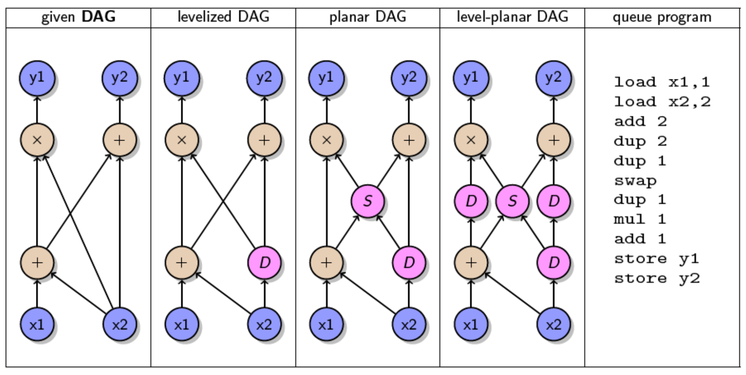
- **universal SCAD machine**
  - one universal SCAD PU
    - operands *I1* and *I2*
    - opcodes *f* for add,sub,mul,...
    - number of result copies *cps*
- b**readth-first traversal** of expression ensures right order of operands in the queues

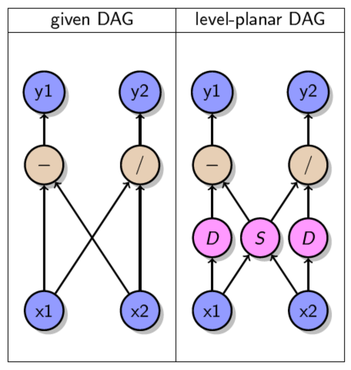


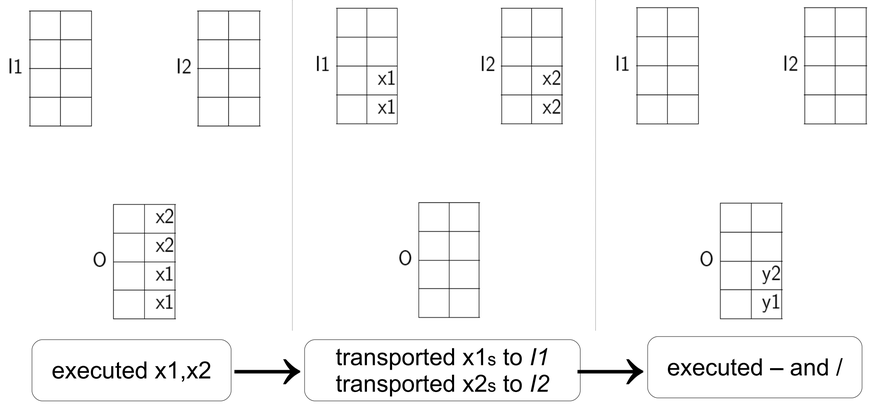
register-less in-order execution → buffer size not limited by the instruction set

Compilation Similar to Queue Machines









Queue Machine

- single queue to hold
  - operands for execution
  - result of execution
- Turing-complete
- code generation by **breadth-first traversal** of expressions although current compilers rely on depth-first traversal for efficient register mapping

Queue Code for DAGs

- levelize directed acyclic graph (DAG) by duplication (*D*) operations, planarize levelized DAG by swap (*S*) operations at edge crossings and relevelize planar DAG by again using duplication operations
- do a consistent left to right (right to left) traversal of level-planar DAG to obtain the queue program
- minimize **swap (*S*) and duplication (*D*) overhead** to obtain optimal queue program
- every queue instruction can be translated to a set of SCAD move instructions to obtain SCAD programs from queue programs

SCAD Code for DAGs – Less Overhead

- SCAD program derived by translating the optimal queue program is not necessarily optimal
- for the example DAG above
  - a queue machine requires 2 duplication and 1 swap operations
  - the universal SCAD machine does not require any swap or duplication operations
- reason: single queue of the queue machine is split into several queues in SCAD
- SCAD machines with many PUs offers even more freedom to schedule instructions
- overhead in terms of number of swap and duplcation operations is further reduced
- also important to analyze control flow boundaries in programs to minimize overhead

SCAD

- **exposed datapath application-specific architecture**
  - bypass registers
- **queue-based code generation**
  - eliminate register usage
  - improves ILP
  - improves timing-predictability
  - however not optimal

Ongoing Research

- **optimal SCAD code generation for DAGs**
  - NP-Complete? Heuristics!
- **buffer size analysis**
- **efficient interconnection networks**
- **out-of-order execution and branch prediction in SCAD**
- **comparison with superscalar and VLIW processors**
- **hardware cost/performance/timing predictions**

Anoop Bhagyanath & Tripti Jain
Chair for Embedded Systems
Department of Computer Science
University of Kaiserslautern

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

(es)
https://es.cs.uni-kl.de