

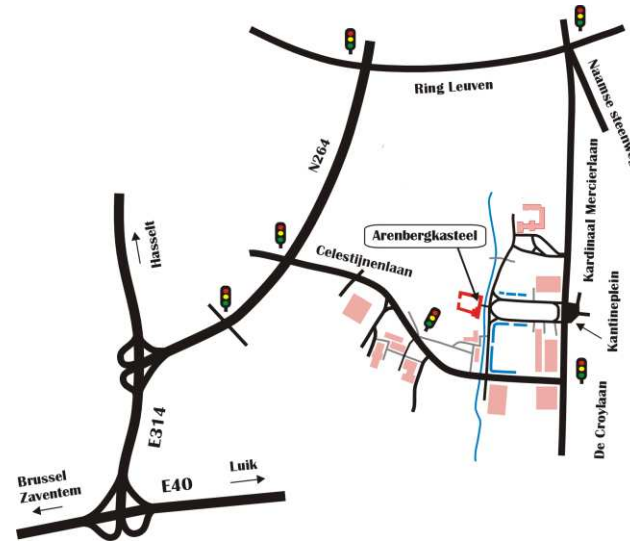
Directions to Arenbergkasteel

Driving Directions

- From the highway
 - Leuven is at the junction of the E40 highway (Brussel-Leuven-Luik-Aachen-Koln) and the E314 highway (Leuven-Hasselt-Genk-Maastricht-Aachen). Leave the E314 at exit 15. You are now on the N264 direction Leuven.
 - At the second traffic lights, turn right. You are now on the Celestijnenlaan (see map below).
 - The defense is at the "Arenbergkasteel". Follow the Celestijnenlaan to the end and turn left. You are now on "De Croylaan". At the "Kantineplein" turn left to the "Arenbergkasteel".
- From Leuven
 - At "Naamsepoort" take the "Kardinaal Mercierlaan" to the "Kantineplein". Turn right to the "Arenbergkasteel".

By Train/Bus

- At the Leuven railway station, take bus line 2 direction "Campus". Leave the bus at the stop "Kantien" in front of the restaurant "Oude Kantien" at the "Kantineplein". The Arenberg Castle is visible from the "Kantineplein".



KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
Celestijnenlaan 200A, B-3001 Leuven—België

Invitation

Mini-symposium

Recent Trends in Compiler Construction

and

PhD Defense

Sven Verdoolaege

Monday April 25th 2005

Schedule

- 14.00 **System design using Kahn Process Networks, the Compaan/Laura approach**
Bart Kienhuis, Assistant Professor, Computer Systems Group, Leiden University, the Netherlands.
- 15.00 Coffee break
- 15.30 **Lattice-based Memory Allocation**
Alain Darte, CNRS Research scientist.
- 16.30 Coffee break
- 17.00 **Incremental Loop Transformations and Enumeration of Parametric Sets**
Sven Verdoolaege, K.U.Leuven
- 19.00± Reception

Location

Arenbergkasteel
Kasteelpark Arenberg 1
3001 Heverlee
Belgium

Registration

The activity is free of charge, but please register before Monday April 18th at the site below.

<http://www.cs.kuleuven.ac.be/~sven/wog/symposium.php>

Abstracts

System design using Kahn Process Networks, the Compaan/Laura approach

High-performance and domain specific embedded architectures, composed of microprocessors, memories, and a number of dedicated coprocessors are very hard to program. On these architectures, applications will be executed that belong to the domains of (real-time) multi-media processing, mobile communication, encryption, or adaptive array processing. On the architectures, instruction level parallelism can be used effectively but not task-level parallelism. To exploit

task-level parallelism, we use the Kahn Process Network model of computation. In the presentation, we will explain the Kahn Process Network model in more detail and indicate why this model is interesting for System Level Design of stream-oriented applications. We present our design methodology based on the Compaan/Laura compilers and present a case in which we convert an M-JPEG application into a Kahn Process Network description and map some of the processes on a CPU and other processes on a FPGA.

Lattice-based Memory Allocation

We investigate the problem of memory reuse in order to reduce the memory needed to store an array variable. We develop techniques that can lead to smaller memory requirements in the synthesis of dedicated processors or to more effective use by compiled code of software-controlled scratchpad memory.

Memory reuse is well-understood for allocating registers to hold scalar variables. Its extension to arrays has been studied recently for multimedia applications, for loop parallelization, and for circuit synthesis from recurrence equations. In all such studies, the introduction of modulo operations to an otherwise affine mapping (of loop or array indices to memory locations) achieves the desired reuse.

We develop here a new mathematical framework, based on critical lattices, that subsumes the previous approaches and provides new insight. We first consider the set of indices that conflict, those that cannot be mapped to the same memory cell. Next we construct the set of differences of conflicting indices. We show a correspondence between a valid modular mapping and a strictly admissible integer lattice – one having no nonzero element in common with the set of conflicting index differences. The memory required by an optimal modular mapping is equal to the determinant of the corresponding lattice. The memory reuse problem is therefore reduced to the (still interesting and nontrivial) problem of finding a strictly admissible integer lattice of least determinant. We then propose and analyze several practical strategies for finding strictly admissible integer lattices, either optimal or optimal up to a

multiplicative factor, and hence for finding memory-saving modular mappings. We explain and analyze previous approaches in terms of our new framework.

Incremental Loop Transformations and Enumeration of Parametric Sets

The geometrical model is a powerful tool for program analysis and optimization and forms the basis on which we build the two parts of this dissertation, a methodology for incremental loop transformations and an efficient enumeration technique for parametric integer sets.

Power consumption for typical embedded multi-media applications is dominated by the storage of and the access to the large multi-dimensional arrays they manipulate. It is now well known that a design methodology for reducing power consumption and improving system performance should apply global loop transformations for increasing locality and regularity of data accesses. In the first part of this dissertation, we propose a two-step global loop transformation approach consisting of a linear transformation focusing mainly on regularity, and a translation focusing on locality. We further develop a refined regularity criterion and show how to perform the translation step incrementally, allowing multiple complicated cost functions to be evaluated.

Many compiler optimization techniques depend on the enumeration of parametric integer sets defined by linear equations. In the second part of this dissertation, we present the first implementation of Barvinok's algorithm applied to the enumeration of parametric polytopes, extending an earlier implementation of this algorithm for a subclass of the enumeration problems we consider, and providing a significant improvement over another implementation based on a different technique. The resulting enumerator may be obtained as an explicit function or as a generating function. We further show that these two representations are polynomially interconvertible and we discuss some approaches for handling generalized enumeration problems.