VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kansliets noteringar Kod	Dnr
2009-11863-70101-37	

2009
Project Research Grant

Area of science
Natural and Engineering Sciences
Announced grants
Research grants NT 15 april 2009
Total amount for which applied (kSEK)

2010	2011	2012	2013	2014
843	868	892		

APPLICANT

Name(Last name, First name)
Sheeran, Mary

Date of birth
590310-2266

Gender
Female

Email address
ms@chalmers.se

Academic title
Professor

Position
Professor

Phone
+46 31 772 1013

Doctoral degree awarded (yyyy-mm-dd)
1984-02-20

WORKING ADDRESS

University/corresponding, Department, Section/Unit, Address, etc.
Chalmers tekniska högskola
Computer Science and Engineering
Software Engineering and Technology

41296 Göteborg, Sweden

ADMINISTERING ORGANISATION

Administering Organisation
Chalmers tekniska högskola

DESCRIPTIVE DATA

Project title, Swedish (max 200 char)
Kontextberoende programgenerering: en tillämpning av funktionell programmering

Project title, English (max 200 char)
Context-aware program generation: an application of functional programming

Abstract (max 1500 char)
Software development is often today a costly, tricky hand-craft, and all the more so in cases where high performance is required. We would like to raise the level of abstraction at which software developers work, freeing them from the tyranny of details that would be better left to automated tools. But in high performance applications, the developer must still, somehow, be given control of those details that are vital to the performance of the resulting code. The need to increase productivity without sacrificing control over fine details in the result is an apparent contradiction. We have observed it in many applications, ranging from wire-aware hardware design, through graphics processor programming to the development of digital signal processing (DSP) software for telecoms base-stations. This project will develop search-based program generation methods that increase programmer productivity and give high performance results.

Abstract language
English

Keywords

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2009-11863-70101-37

Name of Applicant
Sheeran, Mary

Date of birth
590310-2266

Research areas
Computer Science
Review panel
NT-S
Classification codes (SCB) in order of priority
160100, 160102, 160601
Aspects

Continuation grant
Application concerns: New grant
Registration Number:
Application is also submitted to
similar to: **identical to:**

HUMAN AND ANIMAL STUDIES

Human studies
No approved Human studies.
Animal studies
No approved Animal studies.

EQUIPMENT APPLIED FOR IS ALSO TO BE USED IN THE FOLLOWING RESEARCH PROJECT

Research project 1 **Funding source**

Project title and applicant

OTHER CO-WORKER

Name(Last name, First name) **University/corresponding, Department, Section/Unit, Addressetc.**
,
Date of birth **Gender**
Academic title **Doctoral degree awarded (yyyy-mm-dd)**

Name(Last name, First name) **University/corresponding, Department, Section/Unit, Addressetc.**
,
Date of birth **Gender**
Academic title **Doctoral degree awarded (yyyy-mm-dd)**

Name(Last name, First name) **University/corresponding, Department, Section/Unit, Addressetc.**
,

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2009-11863-70101-37

Name of Applicant
Sheeran, Mary

Date of birth
590310-2266

Date of birth **Gender**

Academic title **Doctoral degree awarded (yyyy-mm-dd)**

Name(Last name, First name) **University/corresponding, Department, Section/Unit, Addressetc.**

,

Date of birth **Gender**

Academic title **Doctoral degree awarded (yyyy-mm-dd)**

ENCLOSED APPENDICES

A, B, C, N, S

APPLIED FUNDING: THIS APPLICATION

Funding period (planned start and end date)
2010-01-01 -- 2013-01-01

Staff/ salaries (kSEK)

Main applicant	**% of full time in the project**	**2010**	**2011**	**2012**	**2013**	**2014**
Mary Sheeran	25	281	291	301		
Other staff						
Doctoral student	80	392	407	421		
travel and invited researchers		120	120	120		
computers and computer support		50	50	50		
Total, salaries (kSEK):		843	868	892		

Scientific equipment < 170 kSEK, materials, other costs (kSEK)	**2010**	**2011**	**2012**	**2013**	**2014**
Total, other costs (kSEK):					

Scientific equipment between 170 kSEK and 2000 kSEK (kSEK)	**2010**	**2011**	**2012**	**2013**	**2014**
Total, equipment (kSEK):					

Total amount for which applied (kSEK)

2010	2011	2012	2013	2014
843	868	892		

ALL FUNDING

Other VR-projects (granted and applied) by the applicant and co-workers, if applic. (kSEK)

Proj.no.(M) or reg.nr.	**Funded 2009**	**Funded 2010**	**Applied 2010**
2006-4687	675		

Project title
A functional programming based approach to circuit design and verification challenges

Applicant
Mary Sheeran

Funded 2009	**Funded 2010**	**Applied 2010**
		3453

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2009-11863-70101-37

Name of Applicant
Sheeran, Mary

Date of birth
590310-2266

Project title
Putting functional programming to work (rambidrag application (ICT))

Applicant
John Hughes

Funded 2009 **Funded 2010** **Applied 2010**
603

Project title
Method to limit Nanoscale Circuit Variability

Applicant
Per Larsson-Edefors

Funded 2009 **Funded 2010** **Applied 2010**
452

Project title
ID proj. Domain-Specific Language for Real-Time Embedded Software

Applicant
Anders Persson, Ericsson

Funded 2009 **Funded 2010** **Applied 2010**
657

Project title
IFA prof. Functional programming for improving development of digital signal processing algorithms

Applicant
E. Axelsson

Funds received by the applicant from other funding sources, incl ALF-grant (kSEK)

Funding source **Total** **Proj.period** **Applied 2010**
SSF 1549 2009-2010
Project title
Research Visit to Ericsson, Programming Languages for DSP

Applicant
Mary Sheeran

Funding source **Total** **Proj.period** **Applied 2010**
Intel 398 2009
Project title
New abstraction methods to face challenges caused by

Applicant
Mary Sheeran, Koen Claessen, Per Larsson-Edefors

Funding source **Total** **Proj.period** **Applied 2010**
Ericsson 1000 2009
Project title
DSL4DSP (funding for postdoc)

Applicant
Mary Sheeran

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2009-11863-70101-37

Name of Applicant
Sheeran, Mary

Date of birth
590310-2266

POPULAR SCIENCE DESCRIPTION

Popularscience heading and description (max 4500 char)

Det är mot dagens radio-basstationer vi kopplar upp oss när vi använder våra mobiltelefoner och bärbara datorer. Mjukvaran som utför stationernas signalbehandling (eng. Digital Signal Processing, DSP) är mycket svår att skriva. Programmeraren måste tänka på klockcykler och hur minnesaccesser ska optimeras. Hon måste också se till att tillfullo utnyttja de specialiserade DSP-processorer som tar hand om dom tunga beräkningarna. Således blir denna mjukvaruutveckling både långsam och kostsam. Ericsson är mycket framgångsrika både vad gäller utveckling av sådan mjukvara och att sälja basstationer. Vi skulle vilja förbättra denna mjukvaruutveckling genom att ge programmeraren ett bättre programmeringsspråk att arbeta i. Men på samma gång vill vi inte ge upp den höga prestanda som gör att basstationerna fungerar (och säljs) så väl. Detta är den stora forskningsutmaningen. Hur kan vi utveckla programmeringsspråk och tillhörande kod-genereringsmetoder som låter användaren arbeta på en högre nivå samtidigt som de ger den kontroll över detaljer som behövs för att uppnå hög prestanda? Vi vill utveckla nödvändig programmeringsteknologi för att lösa detta dilemma. I synnerhet kommer vi använda nya idéer om att söka efter program som är anpassade till en viss omgivning, och därmed fungerar effektivt. Vi kommer att demonstrera våra metoder på telekom-programmering i ett projekt gemensamt med Ericsson. Vi kommer också visa hur våra metoder kan generaliseras till andra applikationsområden. Det långsiktiga målet är att lösa det svåra problemet med att programmera datorer som innehåller väldigt många processorer (100- eller 1000-tals istället för 8 eller 16). Sådana datorer kommer finnas, vare sig vi vill det eller ej. Att förstå hur man ska programmera dessa är en fascinerande forskningsutmaning.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2009-11863-70101-37

Name of applicant
Sheeran, Mary

Date of birth
590310-2266

Title of research programme
Context-aware program generation: an application of functional programming

Appendix A

Research programme

Vetenskapsrådet, SE-103 78 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Context-aware program generation: an application of functional programming

The problem that we want to solve

Software development is often today a costly, tricky hand-craft, and all the more so in cases where high performance is required. We would like to raise the level of abstraction at which software developers work, freeing them from the tyranny of details that would be better left to automated tools. But in high performance applications, the developer must still, somehow, be given control of those details that are vital to the performance of the resulting code. The need to increase productivity without sacrificing control over fine details in the result is an apparent contradiction. We have observed it in many applications, ranging from wire-aware hardware design, through graphics processor programming to the development of digital signal processing (DSP) software for telecoms base-stations. This project will develop search-based program generation methods that increase programmer productivity and give high performance results.

State of the Art and Preliminary Results

In programming of both digital signal processors and graphics processors, the programmer typically needs to be aware of the underlying architecture in order to achieve high performance. Domain specific languages (DSLs) have proved suitable in cases where one wants to give the user fine control, with the restriction to a specific domain being what makes this feasible. Examples in hardware design include work at Chalmers on Lava, a DSL for hardware netlist generation [2] and Wired, which enables wire-aware low level hardware design [1].

DSLs for hardware design Lava is a system that supports the design and verification of circuits [2]. It is an extensible domain specific language embedded in the standard functional programming language Haskell. Lava descriptions encode standard ways to build circuits (*connection patterns*) as higher order functions. The standard Haskell function `map` corresponds to placing a component on each element of a list of inputs (a bus).

For example, an important pattern is parallel prefix or scan. Given inputs $[x_0, x_1 \ldots x_{n-1}]$, the prefix problem is to compute each $x_0 \circ x_1 \circ \ldots \circ x_j$ for $0 \leq j < n$, for $\circ$ an associative, but not necessarily commutative, operator. In a construction attributed to Sklansky, one can perform the prefix calculation by first, recursively, performing the prefix calculation on each half of the input, and then combining (via the operator) the last output of the first of these recursive calls with each of the outputs of the second, see Figure 1. To express the construction in Lava, we make use of two connection patterns. `two :: ([a] -> [b]) -> [a] -> [b]` applies its component to the top and bottom halves of the input list, concatenating the two sub-lists that result from these applications. Thus, `two (sklansky plus)` applied to `[1..10]` gives `[1,3,6,10,15,6,13,21,30,40]`. Left-to-right serial composition is written as infix `->-`. The final step (implemented by the function `sfan`) is to combine the last output of the first recursive call (`15` in the example) with each element of the second.

```
sklansky op [a] = [a]
sklansky op as = (two (sklansky op) ->- sfan) as
  where
    sfan as = a1s ++ a2s'
      where
        (a1s,a2s) = splitAt ((length as + 1) `div` 2) as
        a2s'      = [op(last a1s,a) | a <- a2s]

*Main> simulate (sklansky plus) [1..10]
[1,3,6,10,15,21,28,36,45,55]
```

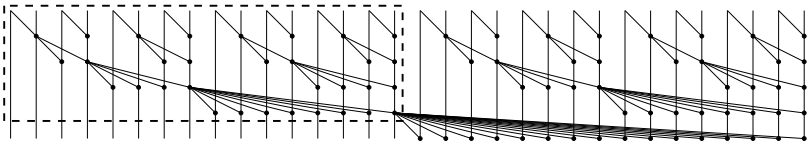

Fig. 1. The Sklansky construction for 32 inputs, illustrated using a diagrammatic notation for prefix networks. It recursively computes the parallel prefix for each half of the inputs; the dotted box shows the first of these recursive calls. It then combines the last output of that call with each of the outputs of the other recursive call.

Lava is a *staged* language, in that Lava descriptions like this are *run* (in this case after choice of operator and input size); symbolic simulation is thus used to generate an internal representation of the circuit, which can then be written out in various ways, and passed to synthesis and formal analysis tools. This pattern of program or circuit generation is typical of *embedded* domain specific languages [11, 5]. Wired can be viewed as an extension of Lava that describes not only logic function (as technology mapped netlists) but also cell placement and some aspects of wiring.

Also based on a functional DSL, Intel's own work on the IDV system (Integrating Design and Verification, [15]) provides a good demonstration of the value of such an approach. It has been shown to enable design exploration, while maintaining correctness via local transforms and associated formal verification. In high performance processors, where design methods are pushing the limits of the technology, providing a system that really supports rather than hinders the designer has demanded a good understanding of which details should be taken care of by automated tools, and which should be left under the control of the designer. Wired was developed in collaboration with Intel, and we have a current project to develop a new DSL and abstraction methods to enable early estimation of power and performance from high level microprocessor models.

DSLs for GPU programming Graphics processing units (GPUs) have moved from being specialised graphics engines to being suitable to tackle applications with high computational demands. For a recent survey of the hardware, programming methods and tools, and successful applications, the reader is referred to [7]. Unlike for current multicore machines, the question for GPUs is how to keep many small processors busy. We are investigating a combinator-based approach to the kind of data-parallel programming that this demands,

via a DSL called Obsidian that currently generates CUDA code (NVIDIA's data-parallel variant of C). In Obsidian, the Sklansky example is written

```
sklansky op 0 = Pure id
sklansky op n = two (sklansky op (n-1)) ->- Pure (fan op) ->- sync

fan op arr = conc (a1, (mapArray (op c) a2))
   where (a1,a2) = halve arr
         c       = a1 ! (fromIntegral (len a1 - 1))
```

A small part of the generated CUDA code is

```
sm1[tid] = ((unsigned int)((((tid & 0xffffff0f) < 8) ?
((int)(sm2[tid])) :
(((int)(sm2[((tid & 0xf0) | 0x7)])) + ((int)(sm2[tid])))))));
__syncthreads();
sm2[tid] = ((unsigned int)((((tid & 0xffffff1f) < 16) ?
((int)(sm1[tid])) :
(((int)(sm1[((tid & 0xe0) | 0xf)])) + ((int)(sm1[tid])))))));
__syncthreads();
```

and there are six further such blocks of three lines plus a synchronising operation (three on either side of these two). Each kernel has a particular thread-id (`tid`) and works on the calculation of one element of the eventual output array. This illustrates our aim of writing higher level descriptions using combinators like `two` and generating bit-twiddling code that gives good performance on the GPU.

The generation is done by the applicant's doctoral student J. Svensson, co-supervised by K. Claessen. It makes advanced use of type classes and arrows in Haskell, and also uses Boolean simplification. This work is very recent and has not yet been published. We regard it as very promising. For a description of early work on Obsidian and a survey of GPU programming languages, see [22]. The most closely related work that we have found is GPUGen [13], which in turn builds upon recent work on Nested Data Parallel programming in Haskell [3]. Our work is distinguished by its aim to *both* raise the level of abstraction *and* aim for high performance. Functions like scan, whose implementation we want to explore, are primitives in other approaches that aim to raise the level of abstraction at which programmers work. However, we also feel that ideas related to fusion in Data-Parallel Haskell will be applicable in both our work on GPU programming and in the DSL for DSP described in the following section. Our expected application area is currently library functions in GPGPU and graphics programming, with widely used examples being sorting, stream reduction and scan. A typical Obsidian user may well write such frequently used functions that are needed in many variants in Obsidian, while writing other critical code directly in CUDA or OpenCL. We have local access to graphics and GPU programming expertise in Ulf Assarsson and his group, and the development of Obsidian is much influenced by our discussions with this group.

A DSL for DSP and related work Since the beginning of 2009, we have been working on moving these ideas (embedded DSLs and combinators) over to the development of the kind of hardware-like software that is used in the DSP parts of baseband processing

in radio base stations. We are developing a domain specific language for the design and implementation of the DSP algorithms that are the core of the signal processing – the computational kernels that consume most of the cycles. These algorithms form the *data-path* of the processing, as distinct from the *control part*, which repeatedly calls them with widely varying parameters, and with very high demands on throughput. The DSL4DSP project involves the Functional Programming Group at Chalmers and ELTE University, Budapest. Ericsson funds one Chalmers postdoc (Axelsson, who developed Wired), the ELTE group and several Ericsson participants with expertise in both research and product development. The intention is that the resulting language be open source, and used and supported by DSP vendors, so this is an ambitious project. We will not succeed unless we generate high performance code. The Swedish Foundation for Strategic Research (SSF) funds the applicant's participation under a Mobility initiative [16], so that she is working 80% on the project at Ericsson during 2009, and will have some "repatriation" funding during 2010. One of the main aims of this application is to tackle the longer term and less domain-specific research questions arising from this work. Axelsson will also apply for an IFA-project from VR to continue the work at Ericsson. A separate Ericsson-internal project is developing an embedded DSL for the control part of the processing. That work will be the subject of an industrial doctoral student (ID) application to VR (from A. Persson at Ericsson).

The DSL builds upon ideas in Lava and Wired. It is purely functional and again staged, but this time the internal representation is a much more sophisticated graph structure. It is no longer sufficient just to unroll the combinators as in Lava, for example. There are considerable possibilities for optimisation, based on ideas such as stream fusion [4], and this will be a major focus of the above-mentioned IFA application. (The ELTE group is building the backend that generates C (eventually with intrinsics) from the internal graph structure.) The project was also inspired by Galois Inc's Cryptol language for the development and verification of crypto algorithms [14]. The generation of VHDL (for hardware generation) from Cryptol was inspired by the applicant's PhD thesis and early work on retiming [17]. Andy Gill, who did much of this work, is now back in academia and we are planning collaboration.

In the development of DSP signal processing, the final software must be partitioned onto the platform, and current tools do not provide good support for this. The result is that necessary repartitioning (for instance when telecom standards change) is alarmingly time-consuming. In addition, it is not possible to explore different possible partitions, or to exploit possibilities for optimisation. For core algorithms, the programmer should be able to place data in the memory hierarchy with sufficient precision and control. But at the same time, the ideal would be to write platform-independent code, and then use a model of the architecture to guide the generation of platform-dependent code. This problem of portability arises both when one considers a single DSP chip (where even moving to another processor from the same vendor can be difficult) and when one considers the collection of processors and accelerators that makes up a platform. Another layer of complexity arises when some of the processors are multicores. Given the nature of many DSP algorithms, it seems likely that future platforms will include highly data-parallel manycore architectures.

The question is "How can we give the programmer control over the fine details that lead to high performance solutions, while maintaining the goal of writing easily portable code?". We believe that some of the building blocks of a solution can be found in our own earlier work on algorithmic circuit generation and on ways to enumerate and analyse possible solutions, leading to the use of search.

More advanced generation methods In Lava, we have explored the notion of *clever circuits* – circuits that have additional Haskell-level *shadow* parameters carrying non-functional properties, allowing them to adapt to their contexts *during circuit generation* [18]. We demonstrated the method on multiplier reduction trees [19]. In the multiplier reduction trees, the cells can be thought of as being placed initially, and it is only the wiring between them that is chosen during generation. In that work, the context consisted only of the shadow values capturing input delays. One can go further and have the context capture both input and required output delays. One can then enumerate and choose between a large number of possibilities for the entire topology of the network, using its recursive decomposition and dynamic programming. We have had considerable success in doing this for parallel prefix networks. We started with elaborate ways to decompose the problem so that one solved different slices of the circuit separately [20], in an effort to make the problem feasible. We have recently found, however, that one can tackle the entire network (of fixed size but for large number of inputs) by exploiting knowledge of which recursive decompositions are likely to give good (that is small) networks. The resulting generator is pleasingly small and the results improve on known best solutions. We have also linked this search to Wired, to enable search for low power networks. A key point here is that one is making significant use of the host language in writing sophisticated generators; so the fact that the DSLs we study are *embedded* is important.

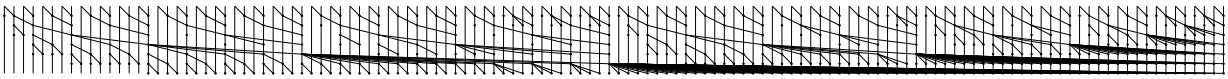

Fig. 2. The new construction for 128 inputs, depth 7. It uses 364 operators, compared to 369 for Ladner Fischer [12] (and 448 for Sklansky).

This work has very recently in turn led to the development of a new parallel prefix algorithm that does not require search, but that grew out of the insights gained from seeing the results of search in many contexts. In the new algorithm, the number of operators (the *size*) for a minimum depth network with $w = 2^n$ inputs approaches $3.5w$, while the Ladner Fischer algorithm approaches $4w$. This is a substantial improvement, different from and also improving considerably on the more advanced of Fich's prefix algorithms [10]; for instance, it requires 14662683 operators for $2^{22} = 4194304$ inputs, while the corresponding sizes for Ladner Fischer and Fich are 16580799 and 16558745 respectively. We would like to investigate ways to exploit the new algorithm. We do not yet know if parallel prefix is a one-off example in which design exploration using search enables the development of new

algorithms. It would be interesting to try to make the approach more systematic and apply it to other standard algorithms.

When charting related work for the DSL4DSP project, we have found research with very similar aims in the SPIRAL project at CMU [9]. The flagship of the project is the SPIRAL program generation system, which, entirely autonomously, generates platform-tuned implementations of signal processing transforms such as the discrete Fourier transform, discrete cosine transform, and many others. This kind of *autotuning* also plays a large role in the Berkeley view of likely developments in parallel programming and architectures [8]. The Berkeley view considers motifs (or dwarfs): related groups of problems, with typical examples being dense linear algebra, combinational logic and finite state machines. SPIRAL is considered to set the standard for library generation for the spectral motif [23]. We are excited by the work on SPIRAL, not only because of its evident success, but also because it is very much in line with our own ideas about the use of search, and is clearly applicable to one of our chosen application areas (DSP algorithms). We expect to borrow ideas from SPIRAL.

What we will do in this project

Clever circuits plus combinators plus search seems to be a very powerful combination. The main aim of this project will be to explore this combination and its application to program generation both for DSP algorithms and for GPU programming.

The planned tasks for the first period of the project are as follows:

1) Continue work on search in algorithm development (prefix). Implement new algorithms in Obsidian. Try to generalise the method to other common data-independent algorithms.

2) Consider the case where the most natural way to express an algorithm is to make modifications (using the clever circuits idea) to one that does more but is very regular. An example is our earlier work on generating median networks from sorting networks [18], where it was possible to reduce the number of comparators needed to produce the median of 25 inputs (a common operation in graphics). This kind of application of clever circuits has not been much explored and is promising. In the DSP area, we will explore its use in the development of sorting networks that are well matched to DSP processors.

3) Model a single DSP processor and explore the use of search to map kernel algorithms to it effectively. This is a good initial test because DSPs display varying degrees of parallelism (both VLIW and replicated ALUs and other accelerators). We also see the possibility (later) of applying these ideas even to the control part and entire platforms. This work should be done in collaboration with others in the DSL4DSP project.

4) Explore ways to find and exploit optimisations when algorithms "meet". A preliminary example is the composition of an FFT, some function `f` and an inverse FFT. One of the main aims of raising the level of abstraction at which the programmers of DSP algorithms work is to enable such optimisations. Here, again, we expect search to play a major role.

We will keep the work concrete through our strong connection to the DSP application area and our contacts with practitioners at Ericsson. The new approach to software generation will be tested on case studies both from baseband and media processing. The intention is also to have Ericsson developers outside the project try out the approach, but this will only happen if we succeed in producing code that is of close to hand-crafted standard.

The longer term aim will be to enable easy partitioning, design exploration and repartitioning, initially for the core algorithms, but later also for the control part. In the first year, the development of associated verification methods will not be a central goal of the project (as existing methods work reasonably well). However, we would later like to experiment with adapting and developing Galois Inc's verification research [6] for this application area. Another source of inspiration will be recent work by Dill and his students [21].

GPU programming will also be a source of case studies. Our view is that the GPU and DSP work will continue to converge (as is already happening in recent work about data-types for vectors or arrays). Longer term work on this project will concentrate on ways to program highly data-parallel systems (100s or 1000s of processors rather than 8 or 16). Building on our expertise in hardware description, we would also like to tackle the issue of power-aware programming (which is forecast to become a central research question if forthcoming telecoms standards are to be widely adopted).

Along with colleagues in the FP group at Chalmers, we will also work on making the development of DSLs more systematic. A central question there is finding data-structures and abstractions that allow details to be moved between levels of abstraction so that working at a higher level can still give high performance results. Both this and the work on context-aware software generation are covered by the multi-project application from the Chalmers FP group.

Collaboration

The link to Ericsson (with the DSL4DSP project and the applied for ID and IFA projects) and the Chalmers Functional Programming Group together provide an excellent research environment for this proposal. Hughes and Sheeran are both PIs in Chalmers application to Vinnova in the strategic research area ICT. This (if funded) should lead to a further strengthening of our research environment. The group has also applied for a multi-project grant in functional programming, DSLs and associated verification methods. We have strong research links to Microsoft Research in Cambridge and to Intel; Satnam Singh and Carl Seger are visiting faculty, interacting strongly with our group. We expect during this project to establish research collaboration with Andy Gill (U. Kansas), with Galois Inc. and possibly with NVIDIA. Locally, we expect to collaborate with Per Stenström in the development of new parallel programming methods.

Impact

Baseband processing is a key component of Ericsson's business. If we succeed in developing a well-functioning embedded DSL for DSP algorithms, we will change the way such algorithms are designed and implemented at Ericsson, reducing development time and cost.

Thus we have already established the route to strong industrial impact. Scientifically, we hope to contribute to solving the pressing problem of how to program parallel machines, aiming particularly for highly data-parallel manycore architectures.

References

1. Emil Axelsson. *Functional Programming Enabling Flexible Hardware Design at Low Levels of Abstraction.* PhD thesis, CSE Dept., Chalmers University of Technology, 2008.
2. P. Bjesse, K. Claessen, M. Sheeran, and S. Singh. Lava: Hardware design in Haskell. In *International Conference on Functional Programming*. ACM Press, 1998.
3. M. M. T. Chakravarty, G. Keller, R. Lechtchinsky, and W. Pfannenstiel. Nepal – Nested Data-Parallelism in Haskell. In *Euro-Par 2001: Parallel Processing, 7th International Euro-Par Conference*, LNCS. Springer, 2001.
4. Duncan Coutts, Roman Leshchinskiy, and Don Stewart. Stream fusion: From lists to streams to nothing at all. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, April 2007.
5. Conal Elliott, Sigbjørn Finne, and Oege de Moor. Compiling embedded languages. *Journal of Functional Programming*, 13(2), 2003. Updated version of paper by the same name that appeared in SAIG '00 proceedings.
6. Levent Erkök and John Matthews. Pragmatic equivalence and safety checking in Cryptol. In *Programming Languages meets Program Verification, PLPV'09, Savannah, Georgia, USA*, pages 73–81. ACM Press, January 2009.
7. J. D. Owens et al. GPU Computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.
8. K. Asanovic et al. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.
9. Markus Püschel et al. SPIRAL: Code generation for DSP transforms. *Proceedings of the IEEE, special issue on "Program Generation, Optimization, and Adaptation"*, 93(2):232– 275, 2005.
10. Faith Ellen Fich. *Two problems in concrete complexity: cycle detection and parallel prefix computation.* PhD thesis, University of California, Berkeley, 1982.
11. Paul Hudak. Modular domain specific languages and tools. In *in Proceedings of Fifth International Conference on Software Reuse*, pages 134–142. IEEE Computer Society Press, 1998.
12. Richard E. Ladner and Michael J. Fischer. Parallel prefix computation. *J. ACM*, 27(4), 1980.
13. S. Lee, M. M. T. Chakravarty, G. Keller, and V. Grover. GPU Kernels as Data-Parallel Array Computations in Haskell. In *Workshop on Exploiting Parallelism using GPUs and other Hardware-Assisted Methods*, 2009.
14. J. R. Lewis and B. Martin. Cryptol: high assurance, retargetable crypto development and validation. In *Military Communications Conference, Volume 2*, pages 820– 825. IEEE, 2003.
15. Carl Seger. Integrating design and verification - from simple idea to practical system. In *Fourth ACM and IEEE Int. Conf. on Formal Methods and Models for Co-Design, MEMOCODE*, 2006.
16. M. Sheeran. Research Visit to Ericsson, Programming Languages for DSP. SM08-0026, SSF, 090101-101231.
17. M. Sheeran. Retiming and slowdown in Ruby. In G.J. Milne, editor, *The Fusion of Hardware Design and Verification*. North-Holland, 1988.
18. M. Sheeran. Finding regularity: describing and analysing circuits that are almost regular. In *Correct Hardware Design and Verification Methods*. LNCS 2860, Springer, 2003.
19. M. Sheeran. Generating fast multipliers using clever circuits. In *Formal Methods in Computer-Aided Design, FMCAD*, volume 3312 of *LNCS*. Springer, 2004.
20. M. Sheeran. Parallel Prefix Network Generation: an Application of Functional Programming. In *Int. Workshop on Hardware Design and Functional Languages, associated with the ETAPS conferences*, 2007.
21. Eric Smith and David Dill. Automatic formal verification of block cipher implementations. In *Formal Methods in Computer-Aided Design (FMCAD)*. IEEE, 2008.
22. Joel Svensson, Mary Sheeran, and Koen Claessen. Obsidian: A Domain Specific Embedded Language for Parallel Programming of Graphics Processors. In *post-symposium proceedings of 20th International Symposium on the Implementation and Application of Functional Languages (2008), in press*, 2009.
23. Samuel Webb Williams. *Auto-tuning Performance on Multicore Computers.* PhD thesis, EECS Department, University of California, Berkeley, Dec. 2008.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2009-11863-70101-37

Name of applicant
Sheeran, Mary

Date of birth
590310-2266

Title of research programme
Context-aware program generation: an application of functional programming

Appendix B

Curriculum vitae

Vetenskapsrådet, SE-103 78 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Mary Sheeran

1. First Degree:

Bachelor of Engineering Science degree, Electrical Engineering, Trinity College Dublin, 1980.

2. Graduate Degrees:

M.Sc. and D. Phil degrees in Computation., Oxford University, 1981 and 1984.

3. Postdoc and visiting positions and fellowship awards:

Visiting post-doctoral researcher, Chalmers (Sweden), 1984-1985.
Visiting Scientist at IBM Almaden Research Center (with John Backus), Summer 1987.
Royal Society of Edinburgh BP Research Fellow, 1989-92.
Senior researcher, Prover Technology AB (part-time), 1997-2003.
Currently working 80% on SSF-funded research visit to Ericsson (Prog. Lang. for DSP)

5. Current position

Professor in computing science, Chalmers University of Technology (75% research) since April 1999. Joined Chalmers as universitetslektor in 1992.

6. Earlier positions

University lecturer in computing science, University of Glasgow, Scotland,1986-1992.
University lecturer in Oxford, 1985-1986
GEC Junior Research Fellow, Programming Research Group, Oxford University, 1983-1984.

7. Parental leave

Parental leave 1985-1986 and 1993-1994 (about 20 months in total)

8. Research supervision to doctorate:

Satnam Singh (1991), Graham Hutton (1992), Koen Claessen (2001), Per Bjesse (2001), Niklas Een (2005), Magnus Björk (2006), Jan-Willem Roorda (2007), Emil Axelsson (2009). Currently supervising Joel Svensson.

Recent research grants

2009-2010 Research visit to Ericsson (SSF mobility scheme)
2009 Abstraction methods in high level hardware design (Intel donation)
2009 A Domain Specific Language for DSP (Ericsson (funds postdoc))
2007-2009 Functional Circuits (VR)
2008 Formal Verification in ASIC Design (Saab Space and NRF)
2006-2007 Performance by Construction (Intel-custom funding from the Semiconductor Research Corporation, industrial liaisons from IBM Austin, and Intel Strategic CAD Labs, Oregon)

Selected professional activities:

Charter member of IFIP Working Group 2.8 (on functional programming).

Steering Committee member for Int. Conference on Formal Methods in Computer Aided Design (FMCAD)

Chair or PC member for Int. Workshop on Designing Correct Circuits, 1990, 1996, 2002, 2004, 2006, 2008.

Co-chair of International Workshop on Hardware Design and Functional Languages (with ETAPS) 2007, PC member 2009.

Co-chair Int. Conference on Formal Methods in Computer Aided Design, 2007.

Extensive programme committee work (e.g. MPC 1992,1995,1998, POPL 1997, CHARME 1999, 2001, 2003, 2005, CAV 2001, FMCAD 1998, 2000, 2002, 2006, TACAS 2001, 2002, DATE 2003, 2004, 2005, CS Russia 2006, WODES 2008, Haskell Symposium 2008, TFP 2009)

External Examiner for doctoral theses at Brunel University and at the Universities of Cambridge, Edinburgh, Glasgow, Kent, York and New South Wales.

Member of examining committee for doctoral theses at Oregon Graduate Institute, KTH, ENS Paris, and Uppsala University and external assessor for doctoral thesis at the Turku Centre for Computer Science.

Leader (with John Hughes) of the Functional Programming Research Group at Chalmers.

Vice chair of VR panel NT-S (Computer Science), 2008.

Chalmers leader and member of Steering Group of DSL4DSP project at Ericsson, 2009-

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2009-11863-70101-37

Name of applicant
Sheeran, Mary

Date of birth
590310-2266

Title of research programme

Appendix C

List of publications

Vetenskapsrådet, SE-103 78 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Selected Publications: Mary Sheeran

Note to non computer scientists Conference articles in computer science are peer reviewed full articles — not 1–2 page abstracts, and are the normal form of refereed publication. The top conferences in each subfield typically have the highest impact factor within that field. All articles listed below are selected for publication by a peer review process, unless otherwise indicated.

Most cited publications (Google Scholar via Harzing's Publish or Perish, duplicates merged)

Sheeran's Hirsch-index is 18 and the following papers are the five most cited.

1. (*) M Sheeran, S Singh and G Stålmarck. Checking safety properties using induction and a SAT-solver. In Proc. Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD), Lecture Notes in Computer Science 1954, Springer, 2000.
   Number of citations: 235.
2. (*) G Jones and M Sheeran Circuit design in Ruby. In *Formal Methods for VLSI Design: IFIP WG 10.5 Lecture Notes*, North-Holland, 1990.
   Number of citations: 202.
3. (*) (**) P Bjesse and K Claessen and M Sheeran and S Singh. Lava: hardware design in Haskell. In Proceedings of the third ACM SIGPLAN international Conference on Functional Programming, ACM Press, 1998.
   Number of citations: 200.
4. M Sheeran and G Stålmarck. A tutorial on Stlåmarck's proof procedure for propositional logic. *Formal Methods in System Design*,16:1, pages 23–58, Springer, 2000.
   Number of citations: 135.
5. (*) (**) M. Sheeran. muFP, a language for VLSI design. In Proceedings of the 1984 ACM Symposium on LISP and Functional Programming, ACM Press, 1984.
   Number of citations: 73.

Journal articles (2002–2009)

6. M. Sheeran. Hardware Design and Functional Programming: a Perfect Match (extended version). In *Journal of Universal Computer Science, JUCS* 11 (7), 2005.
   Number of citations: 16.
7. (*) (**) K. Claessen and M. Sheeran and S. Singh. Using Lava to Design and Verify Recursive and Periodic Sorters. In *Software Tools for Technology Transfer*, Vol. 4, No. 3, May 2003.
   Number of citations: 7.

Articles in refereed collections and conference proceedings (2002–2009)

8. (*) (**) J. Svensson, M. Sheeran and K. Claessen. Obsidian: a Domain Specific Embedded Language for Parallel Programming of Graphics Processors. In *Proc 20th Int. Symposium on the Implementation and Application of Functional Languages*. Springer LNCS, to appear, 2009. (accepted after post-symposium refereeing)
   Number of citations: 1.
9. K. Claessen, N. Een, M. Sheeran and N. Sörensson. SAT-Solving in Practice. In *Proc. 9th International Workshop on Discrete Event Systems*, IEEE, 2008.
   Number of citations: 1.
10. (*) (**) M. Sheeran. Searching for prefix networks to fit in a context using a lazy functional programming language. In *Proc. Int. Workshop on Hardware Design and Functional Langauges (ed. Martin, Seger, Sheeran)*, associated with ETAPS conferences 2007. (acceptance based on peer review of an abstract)
    Number of citations: 5.
11. M. Björk, M. Själander, J. Hughes, M. Sheeran et al. Exposed Datapath for Efficient Computing. In *Proc. HiPEAC Workshop on Reconfigurable Computing*, 2007.
    Number of citations: 1.
12. H. Eriksson, P. Larsson-Edefors and M. Sheeran et al. Multiplier Reduction Tree with Logarithmic Logic Depth and Regular Connectivity. In *Proc. IEEE Intl Symposium on Circuits and Systems (ISCAS)*, IEEE, 2006.
    Number of citations: 4.
13. E. Axelsson, M. Bjrk and M. Sheeran. Teaching Hardware Description and Verification. In *Proc. International Conference on Microelectronic Systems Education*, IEEE, 2005.
    Number of citations: 1.
14. (*) (**) E. Axelsson, K. Claessen and M. Sheeran. Wired: Wire-Aware Circuit Design. In *Proc. Int. Conf. on Correct Hardware Design and Verification Methods (CHARME)*. Springer LNCS 3725, pp. 5–19, 2005.
    Number of citations: 23.
15. M. Sheeran. Hardware design and functional programming: a perfect match (invited paper). In *Proceedings 9th Brazilian Symposium on Programming Languages (SBLP05)*, 2005.
16. (*) (**) M. Sheeran. Generating fast multipliers using clever circuits. In *Proc. Int. Conf. on Formal Methods in Computer-Aided Design, FMCAD04*, Springer LNCS 2312, pp. 6-20, 2004.
    Number of citations: 18.
17. J. Hughes, K. Jeppson, P. Larsson-Edefors, M. Sheeran and P. Stenström and L. J Svensson. FlexSoC: Combining Flexibility and Efficiency in SoC Designs. In *Proc. NORCHIP Conference*, 2003.
    Number of citations: 4.
18. (*) (**) M. Sheeran. Finding regularity: describing and analysing circuits that are almost regular. In *Proc. Int. Conf. on Correct Hardware Design and Verification Methods, CHARME03*, Springer LNCS 2860, 2003.
    Number of citations: 8.

19. K. Claessen, M. Sheeran and S. Singh. Functional Hardware Description in Lava. Chapter in Fun of Programming, Festschrift for Richard Bird, J. Gibbons and O. de Moor (eds.), Palgrave Cornerstones in Computing series, 2002.
    Number of citations: 2.

Other papers, edited proceedings, tech. reports etc. (2002–2009)

20. E. Axelsson, K. P. Subramaniyan and M. Sheeran and P. Larsson-Edefors. Fast Layout Exploration Using the Wired System. Swedish System-on-Chip Conference,2009,to be presented.
21. Koen Claessen, Carl Seger, Mary Sheeran, Emily Shriver and Wouter Swierstra. High level architectural modelling for early estimation of power and performance. In Proc. Int. Workshop on Hardware Design and Functional Languagues, associated with ETAPS, York, 2009. (a short abstract plus presentation)
22. Koen Claessen, Niklas Een, Mary Sheeran, Niklas Sörensson, Alexey Voronov and Knut Åkesson. SAT-solving in practice. Invited submission to special issue of JDEDS journal on WODES workshop, 2008. Under review.
23. J. Baumgartner and M. Sheeran (editors). *Proc. Int. Conf. on Formal Methods in Computer Aided Design*. IEEE Computer Society. ISBN/ISSN: 0-7695-3023-0, 2007.
24. E. Axelsson, K. Claessen and M. Sheeran. Using Lava and Wired for Design Exploration. In *Proceedings of the sixth international workshop on Designing Correct Circuits, Vienna, Mary Sheeran and Tom Melham (editors)*. Workshop associated with the ETAPS conferences, 2006. (acceptance based on refereeing of an abstract)
    Number of citations: 1.
25. M. Sheeran and I. Parberry. A new approach to the design of optimal parallel prefix circuits. Technical Report TR-2006-1, CSE Dept., Chalmers University of Technology, 2006.
    Number of citations: 5.
26. Emil Axelsson, Koen Claessen and Mary Sheeran. Wired - a Language for Describing Non-Functional Properties of Digital Circuits. In *Proc. Int. Workshop on Designing Correct Circuits (DCC)*, associated with ETAPS conferences, 2004. Accepted on basis of short abstract.
    Number of citations: 2.
27. K. Claessen, M. Sheeran and S. Singh. Lava, an Embedded Language for Structural Hardware Design. in *Proc. Designing Correct Circuits* (Sheeran and Melham eds.), Workshop associated with the ETAPS conferences), 2002. Accepted on basis of short abstract.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2009-11863-70101-37

Name of applicant
Sheeran, Mary

Date of birth
590310-2266

Title of research programme

Appendix N

Budget and research resources

Vetenskapsrådet, SE-103 78 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Budget

I apply for a 80% of PhD student salary (the other 20% is covered by teaching) and for 25% of my own salary. I also apply for travel money for me and the PhD student, and also for inviting guest researchers and colleagues from Ericsson for shorter visits. Finally, I apply for money for computer equipment and computer support.

Cost	2010	2011	2012
Mary Sheeran, 25%:	281	291	301
PhD student, 80%:	392	407	421
Travel and invited researchers:	150	150	150
Comp. equip. & support	50	50	50
Total:	873	898	922

I have a VR grant that expires at the end of 2009 (2006-46787). This is a new project application because of the move towards software generation. The DSL for DSP project mentioned in Appendix A is partly funded by Ericsson. In 2009, they paid 1000 kSEK to Chalmers to pay for one postdoc (Axelsson). My particaption in the project has been funded by SSF (SM08-0026, SSF, 090101-101231, 1 548 933 SEK). Of that funding, 549 SEK is for use on my return to Chalmers in 2010, having spent 80% of my time working on the project at Ericsson during 2009.

I currently have a donation of 398 kSEK from Intel. This is being used to pay a stipendiat holder from 1 Feb. 2009. A continuation of this donation is, unfortunately, unlikely because of severe cost-cutting at Intel, though we will try to get continued funding.

I am a PI in Chalmers' application under the government's strategic research initiative in ICT, along with John Hughes, also from the Functional Programming Group. The result of that application will be known in June 2009. If we got this funding, it would increase "faculty money" to Chalmers and would strengthen our research environment.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2009-11863-70101-37

Dnr
2009 -

Name of applicant
Sheeran, Mary

Date of birth
590310-2266

Reg date
2009-04-15 11:52:24

Project title
Context-aware program generation: an application of functional programming

NT

Research grants NT 15 april 2009
Project Research Grant
Computer Science
Chalmers tekniska högskola

NT-S

Appendix S

Signatures

2010	2011	2012	2013	2014	
843	868	892			2010-01-01 -- 2013-01-01

Applicant | **Date**

Head of department at host University | **Clarifi cation of signature** | **Telephone**

A signature on the application is required not only from the applicant but also from the authorised representative of the host university/institution or equivalent (normally the head of the department or establishment where the research is to be conducted). The signature confirms that the department can accommodate the proposed research, position or equipment; that the costing in the application is approved for the department's part, that any proposed experimentation on human or animal subjects has been reported, and that the appplicant has reported any secondary occupations and commercial ties (s)he may have, and nothing inconsistent with good research practice has thereby emerged. The applicant must have discussed these conditions with the representative of the host university/institution or equivalent before the latter approves and signs the application.

Educational Science (not Travel and Conference grants)
Note that the signature of an authorised representative of the host university/institution or equivalent means that the latter undertakes to invest its own resources corresponding to at least one-third of the funds obtained from the Committee for Educational Science, if the application is approved. This co-funding may include existing research if this can be linked with projects in Educational Science. Host universities/institutions or equivalent that are awarded funds must annually submit a special report on how the institution has measured up to the co-financing requirement. For research conducted in collaboration with researchers from other educational institutions, the host university/institution or equivalent is responsible for all agreements with the researchers concerned.

International Postdoctoral Fellowships are administered by the Swedish Research Council. The only signature required on the application is that of the applicant.

Travel, Conference and Publication grants: The only signature required on the application is that of the applicant.

Vetenskapsrådets noteringar
Kod

2009-11863-70101-37

Vetenskapsrådet, SE-103 78 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se