VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kansliets noteringar Kod	Dnr
2011-11863-89113-49	

2011
Project Research Grant

Area of science
Natural and Engineering Sciences

Announced grants
Project research grant NT 13 April 2011

Total amount for which applied (kSEK)

2012	2013	2014	2015	2016
929	938	985	995	

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
031 772 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

Inst. för Data och Informationsteknik
41296 Göteborg, Sweden

ADMINISTERING ORGANISATION

Administering Organisation
Chalmers tekniska högskola

DESCRIPTIVE DATA

Project title, Swedish (max 200 char)
Hårdvaruacceleration av algoritmer genom funktionell programmering

Project title, English (max 200 char)
A functional programming approach to hardware acceleration of algorithms

Abstract (max 1500 char)
The aim of this proposal is to develop methods and tools to enable large scale acceleration of algorithms using reconfigurable hardware (Field Programmable Gate Arrays, FPGAs). FPGAs currently contain resources other than just a fabric of computing elements; examples include fast carry chains, embedded DSP circuits that run much faster than the reconfigurable fabric, and embedded processors. These additional resources make FPGAs into very powerful computing platforms, but also demand sophisticated methods if they are to be efficiently exploited. Previous knowledge from the design of algorithmic blocks for implementation on full custom hardware is not simply transferrable to these augmented FPGAs. A new approach is needed; we propose one based on functional programming and search.

The application areas that we aim to support demand fast data-paths, as distinct from more control oriented computations. Our driving example is fully homomorphic encryption, which was shown to be practical only in 2009. It offers the holy grail of enabling the processing of data without needing to decrypt it. Thus, it could make cloud computing secure. The downside is that current approaches make use of gigantic boolean networks, leading to the need to implement and run such circuits. This application area will provide a tough test of the programming language based circuit design methods that we propose, as well as providing the chance to have real practical impact.

Kod
2011-11863-89113-49

Name of Applicant
Sheeran, Mary

Date of birth
590310-2266

Abstract language
English

Keywords
functional programming, parallel prefix, fully homomorphic encryption, circuit generation, search

Research areas
Computer Science

Review panel
NT-S

Classification codes (SCB) in order of priority
10205, 10206,

Aspects

Continuation grant
Application concerns: New grant
Registration Number:

Application is also submitted to

similar to: **identical to:**

ANIMAL STUDIES

Animal studies
No animal experiments

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.**
,

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)**

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2011-11863-89113-49

Name of Applicant
Sheeran, Mary

Date of birth
590310-2266

ENCLOSED APPENDICES

A, B, C, S

APPLIED FUNDING: THIS APPLICATION

Funding period (planned start and end date)
2012-01-01 -- 2015-12-31

Staff/ salaries (kSEK)

Main applicant	**% of full time in the project**	**2012**	**2013**	**2014**	**2015**	**2016**
Mary Sheeran	25					
Other staff						
Forskarassistent	75	752	777	803	830	
Total, salaries (kSEK):		752	777	803	830	

	2012	**2013**	**2014**	**2015**	**2016**
travel	90	90	90	90	
equipment (computer + FPGA boards)	19		19		
office space	58	60	62	64	
IT-costs	10	11	11	11	
Total, other costs (kSEK):	177	161	182	165	

Total amount for which applied (kSEK)

2012	2013	2014	2015	2016
929	938	985	995	

ALL FUNDING

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

Proj.no.(M) or reg.nr.	**Funded 2011**	**Funded 2012**	**Applied 2012**
2009-4303	1850	1850	
Project title	**Applicant**		
Putting functional programming to work	John Hughes		

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

Funding source	**Total**	**Proj.period**	**Applied 2012**
Ericsson AB	1000	2011	
Project title	**Applicant**		
Feldspar: a domain specific language for DSP algorithm design	Mary Sheeran		

Funding source	**Total**	**Proj.period**	**Applied 2012**
SSF	25000	2011-2016	
Project title	**Applicant**		
RAW FP	John Hughes		

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2011-11863-89113-49

Name of Applicant
Sheeran, Mary

Date of birth
590310-2266

POPULAR SCIENCE DESCRIPTION

Popularscience heading and description (max 4500 char)

Om en funktion eller ett program går för långsamt på en dator kan man snabba upp den genom att implementera delar av programmet på speciell rekonfigurerbar hårdvara. Sådan hårdvara (kallad Field Programmable Gate Array, FPGA, på Engelska) kan programmeras och omprogrammeras om och om igen, för att implementera många olika kretsar, och snabba upp många olika program. Oftast är det en viss del av programmet som väljs ut för att implementeras på FPGA. Det senaste årtiondet har FPGAer blivit allt mer sofistikerade, och nu inkluderar de inte bara den programmerbara logiken utan även extra resurser för att snabba upp vanligt förekommande beräkningar. Tilläggsresurserna kan var så enkla som kretsar som snabbar upp carry-signalerna i en adderare. Eller så kan de vara riktiga processorer som sitter bland den rekonfigurerbara logiken. Sådana turbo-FPGAer är svårprogrammerade, speciellt då algoritmen som skall snabbas upp är dataintensiv. Detta projekt har för avsikt att utveckla nya metoder för att programmera FPGAer, baserade på moderna domänspecifika programspråk, och på smarta sätt att söka efter bra kretsar som skall placeras på FPGA.

För att verifiera våra metoder för programmering av FPGAer skall vi applicera dem på kretsar som behövs för att kryptera data och skydda den från obehöriga som vill stjäla den. Ett stort resultat inom kryptografi (från 2009) indikerar att det faktiskt är möjligt att göra beräkningar med krypterad data utan att först dekryptera den. Om man kan göra den nya krypteringsmetoden tillräckligt snabb så kommer den att ha ett stort genomslag, då den möjliggör säkra beräkningar i molnet (Eng. cloud computing). Nackdelen med den nya sortens kryptering är att den kräver att man konstruerar gigantiska kretsar som en del av beräkningen. Vi kommer att utveckla nya metoder för att designa och implementera sådana stora kretsar. Om vi lyckas i forskningen kommer vi att bidra till ett enormt steg framåt inom datasäkerhet.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2011-11863-89113-49

Name of applicant
Sheeran, Mary

Date of birth
590310-2266

Title of research programme
A functional programming approach to hardware acceleration of algorithms

Appendix A

Research programme

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Appendix A: A functional programming approach to hardware acceleration of algorithms

Mary Sheeran (ms@chalmers.se, +46 31 772 1013)

CSE Dept., Chalmers

Purpose and aims

The aim of this proposal is to develop methods and tools to enable large scale acceleration of algorithms using reconfigurable hardware (Field Programmable Gate Arrays, FPGAs). FPGAs currently contain resources other than just a fabric of computing elements; examples include fast carry chains, embedded DSP circuits that run much faster than the reconfigurable fabric, and embedded processors. These additional resources make FPGAs into very powerful computing platforms, but also demand sophisticated methods if they are to be efficiently exploited. Previous knowledge from the design of algorithmic blocks (such as fast binary adders) for implementation on full custom hardware is not simply transferrable to these augmented FPGAs. A new approach is needed, and we propose one based on functional programming and search.

The application areas that we aim to support demand fast data-paths, as distinct from more control oriented computations. They are medium scale cryptography (which demands Montgomery multiplication and exponentiation), large scale correlators and convolvers for astronomy and, finally, fully homomorphic encryption (FHE), which (currently) demands arithmetic on millions of bits, and which, if successfully made practical, will enable secure cloud computing. The proposed research is distinguished from current approaches by:

- the strong emphasis on data-paths. This leads to the need to have fine control over layout on the FPGA and also makes starting from a sequential C-like description infeasible.
- a programming language based approach to exploiting additional resources (including processors) on the FPGA, combined with the use of search
- the sheer scale of the circuits needed in fully homomorphic encryption
- initial strong results on parallel prefix networks (a key building block in fast adders)

Survey of the Field

FPGA programming

FPGA programming is steadily moving towards greater use of High Level Synthesis, but this is hard to reconcile with having fine control over resource use, particularly for data-path implementation. At ENS Lyon, the FloPoCo project aims to develop high performance floating point cores for FPGAs, and it has found the need to develop new methods to implement even binary adders on FPGAs, particularly with pipelining is required [5]. A number

of authors have considered ways to map chaining computations onto the fast carry chains of FPGAs, enabling implemenation of many more operations that just ripple carry adders (e.g. [7, 14]). Our aims are similar, but we want to put control of the mapping into the hands of the programmer (assisted by the use of search). Researchers currently implementing very large and regular digital signal processing arrays for use in astronomy have found that current design tools make the necessary control of layout on the FGPA both slow and painful. The development of tool-chains for FPGA programming has concentrated on relatively small scale control-oriented applications, and on harnessing *software* developers – with the result that there is much work on C-like languages for FPGA design. This approach is inappropriate for our very high performance, very highly parallel applications.

Domain Specific Languages (DSLs)

Domain specific languages (DSLs) can be used to give fine control of resources, with the restriction to a specific domain being what makes this feasible. Examples in hardware (FPGA) design include our work with Singh and later Claessen on Lava, a DSL for hardware netlist generation [3]. Lava has been influential and there are now several new implementations (for example [10], which provides new abstractions, including ways to describe and control memories). We see strong possibilities for collaboration here. The ability to control geometry is a key ingredient of the success of Lava in real applications (as implemented by Singh at Xilinx). Working with Intel, we explored the control of geometry at an even finer level of detail in work on Wired, which enables wire-aware low level hardware design [2]. This proposal aims to return to work on Lava, and on sophisticated circuit generation methods, and to extend it to take account of developments in FPGAs that now combine the FPGA fabric with other computing resources.

Cost models and dynamic programming

The use of integer linear or dynamic programming has a strong history in arithmetic circuit generation (e.g. the classic work on optimal multipliers [21]). Such methods are closely associated with cost models. In our own approaches to circuit generation, we have relied heavily on Non-Standard Interpretation, a variant on classic abstract interpretation, to estimate costs for the purposes of controlling the search for sufficiently good solutions. We will need to use much more sophisticated cost modelling if we are to succeed in exploiting FPGAs plus extra resources such as embedded DSPs or multipliers. We believe that we will be able to build on work by Blelloch and his co-workers on cost models for parallel functional programming [20]. Although our context is rather different, we see strong parallels between our envisaged cost models and those proposed by Blelloch et al. Both lines of research place a strong emphasis on a programming language (rather than a machine model) based approach to algorithm discovery, mapping and profiling.

Cryptography, our main application area

Modular multiplication is an important building block in modern cryptographic algorithms. Chow et al have recently studied the implementation on FPGA of a recursive Karatsuba-

based Montgomery multiplier [4]. We will start with this recursive construction as one of our case studies as it seems well suited to the use of search-based generation techniques and will force us to adapt those methods to real FPGA generation (see below).

Our main motivating application is, however, fully homomorphic encryption (FHE), which was shown possible (if very difficult) in Gentry's thesis from 2009 [8]. Gentry's result has created an enormous splash. In the US, DARPA has appointed Galois, Inc. as research integrator for the $20m PROCEED program (Programming Computation on Encrypted Data) whose goal is to make it feasible to execute programs on encrypted data without having to decrypt the data first. "If we are successful with PROCEED, it fundamentally changes the calculus for computations in untrusted environments on computer systems of unknown provenance. The potential implications for the cybersecurity of cloud computing architectures are profound" states DARPA Director Regina Dugan in testimony submitted to the US House Subcommittee on Emerging Threats and Capabilities, March 1, 2011. Homomorphic encryption offers the possibility of being able to delegate the *processing* of data, without having to give away *access* to it. Gentry has provided an encryption scheme that "keeps data private, but that allows a worker that does not have the secret decryption key to compute any (still encrypted) result of the data, even when the function of the data is very complex" and that thus "helps make cloud computing compatible with privacy" [9]. The downside is that all known FHE schemes are computationally extremely expensive, as they encode the decryption function (inefficiently) as a circuit, and then, in the *Evaluate* algorithm, replace each bit in that circuit with a large ciphertext that encodes that bit. Much work by many researchers will be needed to make a truly practical FHE scheme. Finding ways to implement and run the very large boolean circuits that result is what interests us, partly because it provides the strictest of tests of our proposed FPGA design methods, and partly because the need to implement very large circuits opens new research questions about key algorithms.

We have strong links to Galois Inc., which works very much with functional programming and domain specific languages. Part of Galois Inc's work on the PROCEED project will use their Cryptol DSL for the development and verification of cryptographic algorithms [13] as a means to demonstrate research results in the project. The generation of VHDL (for FPGA programming) from Cryptol was inspired by the applicant's DPhil thesis and by her early work on retiming [16]. Andy Gill, who did much of this work at Galois, is now back in academia and we are planning collaboration, as he continues to work with his new version of Lava, and with FPGAs for algorithmic computations [10]. While we appreciate the expressiveness that Cryptol's advanced type system brings, we are firmly convinced of the need for an expressive meta-programming layer if we are to meet the enormous challenges of fully homomorphic encryption, which currently involves (in one of its "almost homomorphic" parts) binary arithmetic on 13 million bit numbers.

Preliminary Results

DSLs for hardware design

Lava is a system that supports the design and verification of circuits [3]. 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). Lava includes a way to capture sharing in circuit descriptions, so that one can write what look like plain Haskell descriptions, and do not need to resort to heavier weight constructions (such as monads) when descrbing cyclic circuits.

For non-cyclic circuits, it is easy to describe circuits directly in Haskell, and to define various interpretations of them. 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. The Haskell description is parameterised on a fan structure, which both passes through its first input and applies the binary operator to that input and each of the remaining elements of its input list, to give an output list whose length is the same as that of the input list:

```
mkFan op (i:is) = i:[op i k | k <- is]

pplus = mkFan (+)
```

For example, `pplus [1..8]` gives the list `[1,3,4,5,6,7,8,9]`. The Haskell description of the algorithm contains two recursive calls of `skl`, each operating on roughly half of the input. The fan structure is used to combine the last output of the first of these with each of the outputs of the other:

```
skl _ [a] = [a]
skl f as  = init los ++ ros'
  where
    (los,ros) = (skl f las, skl f ras)
    ros'      = f (last los : ros)
    (las,ras) = splitAt (chalf (length as)) as

chalf n = n - n `div` 2 -- Ceiling of n/2
```

Now, `skl pplus [1..4]` is `[1,3,6,10]` and `skl pplus [5..8]` is `[5,11,18,26]`. The final fan structure applies to `(10 : [5,11,18,26])`, so that the result of `skl pplus [1..8]` is `[1,3,6,10,15,21,28,36]` Figure 2 shows another standard prefix network construction due to Ladner and Fischer [12]. Both of these diagrams were generated by *running* their Haskell descriptions using a suitable building block (parameter) that gathers the necessary information. It is also possible to run circuit descriptions in order to get cost estimations, and this can be done either after, or, more interestingly, during circuit generation.

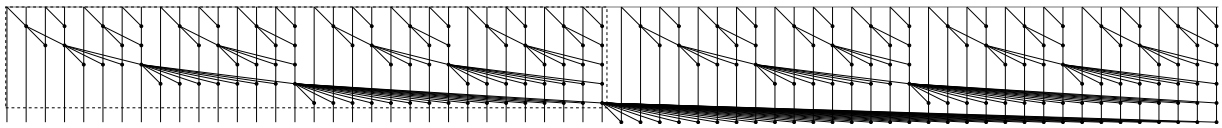

Fig. 1. The Sklansky construction for 64 inputs, illustrated using a diagrammatic notation for prefix networks. It recursively computes the parallel prefix for each half of the inputs; the dotted box shows the first of these recursive calls. It then combines the last output of that call with each of the outputs of the other recursive call.

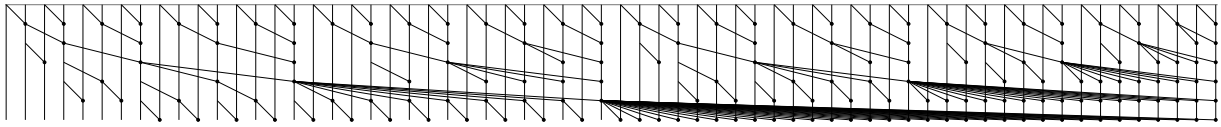

Fig. 2. The Ladner Fischer (LF) construction for 64 inputs. Note how the bottom of the left part of the network makes use of the entire network depth, unlike in the Sklansky construction.

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* [17]. We demonstrated the method on multiplier reduction trees [18]. 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. By choosing a good recursive decomposition, one can find good (that is small) networks, of fixed size but for large number of inputs, using a variety of different measure functions, and including constraints on fanout [19]. The resulting generator is pleasingly small and the results improve on known best solutions for depth size optimal 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.

This work has 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 on the more advanced of Fich's prefix constructions [6]; for instance, it requires 14662683 operators for $2^{22} = 4194304$ inputs, while the corresponding sizes for Ladner Fischer and Fich are 16580799 and 14851947 respectively. Upon the recent appearance of this result in the Journal of Functional Programming, we were contacted by a Russian complexity theorist, who had proven an *exact* lower bound for 2^n-input, depth n parallel prefix networks [15].

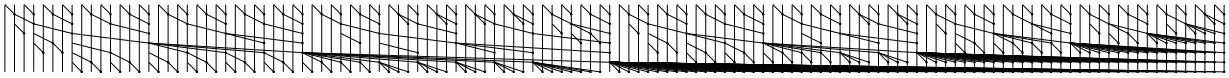

Fig. 3. The new construction for 128 inputs, depth 7. It uses 364 operators, compared to 369 for LF [12] (and 448 for Sklansky).For 256 inputs, the three sizes are 773 for our construction, 792 for LF and 1024 for Sklansky.

Sergeev has proved that the number of operators in a prefix network with 2^n inputs and depth n is bounded below by $(3.5 * 2^n) - (8.5 + 3.5 * (n\%2)) * 2^{(n/2)} + n + 5$, where % is the modulus function. The result relies on a sophisticated argument about the number of redundant operators that must appear in a network of shape similar to the Ladner Fischer construction. (Sergeev's result has so far only been published in preliminary form, and only in Russian. A slightly longer version will appear mid-year and will be translated to English.) The interesting thing is that our construction matches this bound *exactly*, and so is, to the best of our knowledge, the first presentation of an optimal (that is smallest possible) minimal depth prefix network construction. We feel that our success in solving this open problem in prefix network design was due to the use of functional programming plus search, and to the ease of experimentation that ensued. Our aim, now, is to develop this approach further to enable the implementation both of very large prefix networks and of other key algorithms such as sorting and median finding. This will involve both further work on the algorithms themselves and on ways to implement these algorithms on FPGAs.

Our concentration on prefix networks grew out of contacts at Intel Strategic CAD Labs, and has its basis in the fact that prefix networks are key elements both in arithmetic ciruits (where they provide a means to implement fast carry propagation in adders) and more generally in microprocessors, where, for example, they are used to implement priority encoders. There is much work on parallel prefix networks for VLSI circuits, but this does not transfer easily to FPGAs because of the additional resources, particularly fast carry chains. This is well explained in reference [1], which points out that there has been no systematic study of parallel prefix networks on FPGAs, and considers the problem of implementing larger prefix networks (up to 256 inputs) with word level rather than bit level operators. However, one of the chosen implementations is labelled Ladner Fischer, but is actually the Sklansky construction (a common misconception in papers and even text books). We speculate that the cause of this widespread misconception is a lack of suitably expressive programming tools to describe slightly irregular algorithms, and of associated tools to map those algorithms to the FPGA fabric, including the specific extra computing resources that the FPGA provides. At 256 inputs, implementing Sklansky rather than Ladner Fischer means using 1024 instead of 792 operators, a difference that would probably affect power consumption (which tends to depend on circuit size). One of our aims in this project is conduct a serious study of parallel prefix network implementation on FPGAs, first for medium scale (up to say 1024 inputs), and later for the gigantic prefix networks that will be needed for arithmetic on very large numbers, as required in fully homomorphic encryption.

Project plan

Planned tasks in the first phase of the project

Circuit needs of current cryptography Study the need for circuits in current crypto, with emphasis on Montgomery multiplication and exponentiation. Use these as guiding case studies (along with prefix networks) in the following tasks.

Making search more systematic Continue work on search in algorithm development. Make the approach more systematic. Develop combinators for search. Extend the notion of context to include richer constraints on the solution (e.g. the existence of carry chains that should be used). This will entail developing much more sophisticated cost models than those used during our work on circuit geneation so far.

First practical steps in exploiting and controlling additional computing resources: fast carry chains Working with real FPGA implementations, develop the above-mentioned search methods for exactly the case of fast carry-chains. With Satnam Singh, perform a comparative study of parallel prefix network implementation on FPGA.

Result at the end phase one (18 months) Serious study of medium scale prefix networks and Montgomery multipliers on FPGAs, with strong emphasis on programmer control of resources. This will include higher radix prefix networks, where results on optimality are lacking, so that work on the algorithms themselves is needed. This will form the basis for later work on implementing very large scale prefix networks.

Planned tasks in the second phase of the project

Clever circuits revisited 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 [17], where it was possible to reduce the number of comparators needed to produce the median of 25 inputs (a common operation in graphics) from 99 to 96. This kind of application of clever circuits has not been much explored and is promising.

Further work on exploiting and controlling additional computing resources: DSPs and processors Extend work on ways to express use of computing resources + FPGA routing fabric. First consider DSPs (which provide fast arithmetic), and then processors. The need to make use of processors will place new demands on our algorithm decomposition methods, as one must now decide which sub-parts to implement in the processors. Our approach to this problem will be to develop programming oriented cost models, very much along the lines of those developed for parallel functional programming by Blelloch and his coworkers [20]. In the search-based prefix network generation described earlier, the context for the search is something akin to a hole that indicates position and delay on inputs and outputs, and into which the final network must fit. Now, the context is going to have to contain additional resources placed within the hole. The combinators for expressing the search will

have to be more sophisticated, to give control over placement of functions on the FPGA fabric or on the additional computing resource.

The main area of application: fully homomorphic encryption Study fully homomorphic encryption and its need for very large arithmetic circuits. This application will force us to develop ways to decompose the algorithms and to play with space time trade-offs. For the extraordinary demands imposed by FHE, it may be necessary to consider new FPGA structures (along the lines of work by Hauck et al [11]).

Expected results at the end of the project

- Greater understanding of data-independent (circuit-like) parallel prefix algorithms both in theory and with regard to practical implementation at very large scale.
- Demonstration of implementations of very large scale arithmetic circuits (with parallel prefix as a key component) for use in FHE.
- Methods of programming combinations of FPGAs with other computing resources. The methods of programming FPGAs + CPUs will give first results in the search for hardware software codesign methods for algorithmic (rather than control oriented) problems.

Once we have the cost models and search combinators that enable easy programming of additional resources on the FPGA, success will be measured both in terms of performance and area of the generated applications, and of ease of programming new applications.

Collaboration

The Chalmers Functional Programming Group provides an excellent research environment for this proposal. The group is very well resourced. Our work in DSLs for software includes development of a DSL framework, which will be very useful in this project.

Singh, with whom we worked on Lava, is now at MSR, Cambridge. He has recently worked on methods for programming of heterogenous systems, and on ways to translate (ordinary) Haskell programs into circuits. He is visiting faculty, interacting strongly with our group. Both his recent research and his practical expertise in FPGA programming will benefit us. We plan to work together on FPGA implementations of cryptographic algorithms, and on the necessary cost models and search-based programming methods. Arvind, from MIT, is working on the use of the Bluespec DSL in hardware-software codesign. Our work is more data-path oriented, but we will doubtless also need some control-oriented aspects, just as the Bluespec work has incorporated circuit design patterns similar to ours. Harper's work with Blelloch on cost models has inspired us. It was John Launchbury, CTO of Galois Inc., who introduced the author to fully homomorphic encryption. The author has the privilege of meeting all of these researchers yearly at the IFIP working group on functional programming, of which she is a member. We expect during this project to establish research collaboration with Andy Gill (U. Kansas). We also plan collaboration with Marc Pouzet and Jean Vuillemin, who, with A. Cohen, are forming a group to work in

synchronous programming at ENS, Paris. We have strong shared interests in algorithms, hardware design and functional programming languages. Mutual visits are planned. Thus, the proposed research will be supported by a strong network of colleagues and collaborators.

Budget and part of project plan, Overlap with existing projects

Sheeran is co-applicant on Hughes' VR frame grant "Putting functional programming to work: Software Design and Verification using Domain Specific Languages". In that proposal, we excluded work on hardware design, in order to keep the proposal focussed. We made the same decision in our recent (granted) proposal to SSF on resource aware functional programming. That proposal also concentrates entirely on software development methods. This proposal is about hardware design methods, and is designed to support the author's work in this area. The proposed research does not overlap with either of the above-mentioned proposals. This proposal aims to fund 100% of the proposed work.

The intention is to hire an assistant professor (forskarassistent) on the project, possibly from among our existing postdocs. Sheeran plans to spend approximately 25% of her time on this work (funded from other sources). The requested funding for equipment includes 20.000 SEK for FPGA boards.

Significance

Programming FPGAs with embedded carry chains is non-trivial and tends to be done on a case by case basis, with specialised module generators. We want to use the *programming language* to make it more possible for the user to control the final result, by building upon a library of search combinators.

The problem of how to program FPGAs with embedded computational resources such as processors is harder. Solving it will enable greater use of such platforms. The problem also bears some relationship to the more general question of how the transition from pure software to hardware-accelerated software should be made, or more generally how to do hardware software co-design. We don't want to attack the whole of that complex and so far alarmingly difficult problem, but to specialise our work to large data-paths, in order to make progress on investigating *simple* approaches. Any success here will have broad impact. The development of usable programming language based methods for expressing ways in which algorithms should be divided between CPUs and hardware accelerators would benefit applications such as baseband signal processing, where accelerators for key functions like Fast Fourier Transform are routinely used. Our methods could also ease the use of FPGA simulation of VLSI circuits in hardware verification (so-called emulation).

We expect to make progress in theoretical questions relating to key algorithms (such as parallel prefix or multipliers) – developing a programming language rather than a machine model oriented approach to this work on complexity results for real circuits. Parallel prefix is a central algorithm in computer science, as it sheds light on parallel implementations of an apparently sequential algorithm. It is also a key building block in data parallel software,

such as that used in graphics processing. Results here will influence not only hardware but also software implementations.

The holy grail, though, is to find principled ways to design and implement the gigantic circuits that are currently known to be needed to achieve a practical form of fully homomorphic encryption. We have contacts with Galois Inc., who are the research integrator in the DARPA PROCEED initiative in this area, so there will certainly be a route to real application of our results, should they merit such application. The author has in recent years prioritised work on software development methods, and has successfully engaged with Swedish industry in that area. However, the combination of hardware design and functional programming continues to fascinate, and this new holy grail provides challenging research questions, highly qualified collaborators and a route to high impact.

References

1. N. Abbas, S. Derrien, P. Quinton, and S. Rajopadhye. Accelerating HMMER on FPGA using Parallel Prefixes and Reductions. In *Proc. IEEE Int. Conf. on Field-Programmable Technology (FPT'10)*. IEEE, 2010.
2. Emil Axelsson. *Functional Programming Enabling Flexible Hardware Design at Low Levels of Abstraction*. PhD thesis, CSE Dept., Chalmers University of Technology, 2008.
3. P. Bjesse, K. Claessen, M. Sheeran, and S. Singh. Lava: Hardware design in Haskell. In *International Conference on Functional Programming*. ACM Press, 1998.
4. Gary C.T. Chow, Ken Eguro, Wayne Luk, and Philip Leong. A Karatsuba-Based Montgomery Multiplier. *Int. Conf. on Field Programmable Logic and Applications*, pages 434–437, 2010.
5. F. de Dinechin, H.D. Nguyen, and B. Pasca. Pipelined FPGA Adders. In *Int. Conf. on Field Programmable Logic and Applications*. IEEE Computer Society, 2010.
6. Faith Ellen Fich. *Two problems in concrete complexity: cycle detection and parallel prefix computation*. PhD thesis, University of California, Berkeley, 1982.
7. Michael T. Frederick and Arun K. Somani. Beyond the Arithmetic Constraint: Depth-Optimal Mapping of Logic Chains in LUT-based FPGAs. In *Symposium on Field Programmable Gate Arrays*. ACM, 2008.
8. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.
9. Craig Gentry. Computing Arbitrary Functions of Encrypted Data. *Communications of the ACM*, March 2010.
10. A. Gill, T. Bull, A. Farmer, G. Kimmell, and E. Komp. Types and type families for hardware simulation and synthesis: The internals and externals of kansas lava. In *Trends in Functional Programming*, 2010.
11. S. Hauck, T.W. Fry, and M.M. Hosler. High-performance carry chains for FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(2), 2000.
12. Richard E. Ladner and Michael J. Fischer. Parallel prefix computation. *J. ACM*, 27(4), 1980.
13. 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.
14. H. Parandeh-Afshar, P. Brisk, and P. Ienne. Exploiting fast carry-chains of FPGAs for designing compressor trees. In *Field-Programmable Logic and Applications*. IEEE, 2009.
15. Igor S. Sergeev. Some complexity estimations for parallel prefix schemes (in Russian). In *Proc. 10th Int. Seminar on Discrete Mathematics and its Applications*. Moscow State University, Feb. 2010.
16. M. Sheeran. Retiming and slowdown in Ruby. In G.J. Milne, editor, *The Fusion of Hardware Design and Verification*. North-Holland, 1988.
17. M. Sheeran. Finding regularity: describing and analysing circuits that are almost regular. In *Correct Hardware Design and Verification Methods*. LNCS 2860, Springer, 2003.
18. M. Sheeran. Generating fast multipliers using clever circuits. In *Formal Methods in Computer-Aided Design, FMCAD*, volume 3312 of *LNCS*. Springer, 2004.
19. M. Sheeran. Functional and dynamic programming in the design of parallel prefix networks. *Journal of Functional Programming*, 21(1), 2011.
20. Daniel Spoonhower, Guy E. Blelloch, Phillip B. Gibbons, and Robert Harper. Space Profiling for Parallel Functional Programs. *Journal of Functional Programming*, 20(5–6), 2011.
21. P. F. Stelling, C. U. Martel, V. G. Oklobdzija, and R. Ravi. Optimal Circuits for Parallel Multipliers. *IEEE Trans. Comp.*, 47(3), 1998.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2011-11863-89113-49

Name of applicant
Sheeran, Mary

Date of birth
590310-2266

Title of research programme
A functional programming approach to hardware acceleration of algorithms

Appendix B

Curriculum vitae

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Mary Sheeran CV

1. First Degree

Bachelor of Engineering Science degree, Electrical Engineering, Trinity College Dublin, B.A. (mathematics), B.A.I. (elec. eng.), 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.
- worked 80% on SSF-funded mobility project, research visit to Ericsson (Prog. Lang. for DSP), 2009-2010. (Project was extended because of 5 month absence in 2009 due to vision problems, now fixed.)

5. Current position

Professor in computing science, Chalmers University of Technology (75% research, 2010) since April 1999. Joined Chalmers as universitetslektor (University Lecturer) 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. Absences

Parental leave 1985-1986 and 1993-1994 (about 20 months in total). Also 5 months absence because of vision problems in 2009.

8. Research supervision to doctorate

- Satnam Singh (1991) Analysis of Hardware Description Languages
- Graham Hutton (1992) Between functions and relations in calculating programs
- Koen Claessen (2001) Embedded Languages for Describing and Verifying Hardware
- Per Bjesse (2001) Gate Level Description of Synchronous Hardware and Automatic Verification Based on Theorem Proving
- Niklas Een (2005) SAT Based Model Checking
- Magnus Björk (2006) A First Order Extension of Stålmarck's Method
- Jan-Willem Roorda (2007) Semantics, Decision Procedures and Abstraction Refinement for Symbolic Trajectory Evaluation
- Emil Axelsson (2008) Functional Programming Enabling Flexible Hardware Design at Low Levels of Abstraction

Above 8 students supervised to doctorate, main supervisor in all cases. Koen Claessen was technical supervisor for Roorda.

5 of the above students obtained lic. before their doctorates. Mia Indrika and Carl-Johan Lillieroth also obtained lic. degrees. I have two current students at Chalmers, Joel Svensson and Anders Persson (VR-funded ID student, employed by Ericsson). I am co-supervisor of Eva Burrows, a PhD student at the Univ. of Bergen. She will defend her thesis on May 23 2011.

Postdocs supervised

- Andrew D. Gordon (Senior Researcher, Microsoft Research, Cambridge);
- Graham Hutton (Professor in Computer Science, University of Nottingham);
- David Cachera (Researcher, ENS Cachan, France);
- Gordon Pace (Senior Lecturer, University of Malta);
- Matthias Sauer (Chief Scientist and Senior Director at Infineon Technologies);
- Ricardo Massa Ferreira Lima (Lecturer, Univ. de Pernambuco, Brazil);
- Emily Shriver (Intel Strategic CAD Labs, Oregon)
- Emil Axelsson (working on Feldspar project)
- Josef Svenningsson (working on Feldspar project)

Recent research grants

- 2011–2016 RAW FP: Resource Aware Functional Programming, SSF frame grant IT 2010 (co-applicant and co-leader)
- 2009–2010 Research visit to Ericsson (SSF mobility scheme)
- 2009-2012: Co-applicant on Software Design and Verication using Domain Specic Languages (VR, multi-project grant in strategic ICT funding the Functional Programming Group at Chalmers)
- 2009 Abstraction methods in high level hardware design (Intel donation)
- 2009–2011 A Domain Specific Language for DSP (Ericsson, funds (part of) 2 postdocs)
- 2007–2009 Functional Circuits (VR)
- 2008 Formal Verification in ASIC Design (Saab Space and NRF)
- 2004–2006 Clever Circuits (VR)
- 2006–2007 Performance by Construction (Intel-custom funding from the Semiconductor Research Corporation, industrial liaisons from IBM Austin, and Intel Strategic CAD Labs, Oregon)
- 2003–2006 Wired: Expressing and Estimating Non-Functional Properties of Digital Circuits (Intel-custom funding from the Semiconductor Research Corporation)

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 (FMCAD), 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, CUFP 2010, IFL 2011, ITSLE 2011)
- 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 and 2010.
- Chalmers leader and member of Steering Group of Feldspar project at Ericsson, 2009-
- Steering Group of Hiperfit (a large Danish funded centre for high performance computing, functional programming and finance, based at Copenhagen University)

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2011-11863-89113-49

Name of applicant
Sheeran, Mary

Date of birth
590310-2266

Title of research programme
A functional programming approach to hardware acceleration of algorithms

Appendix C

List of publications

Vetenskapsrådet, Box 1035, SE-101 38 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 20 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: 344
2. (∗) 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: 289
3. 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: 219
4. 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: 91
5. M. Sheeran and G. Stålmarck. A tutorial on Stålmarck's proof procedure for propositional logic (conference paper version). In Proc. Int. Conf. on Formal Methods in Computer Aided Design, Springer LNCS, 1998. (The journal version of the paper in FMSD is considerably extended and has an additional 80 citations.)
   Number of citations: 85

1. Journal articles (2003–2011)

1. (∗) M. Sheeran. Functional and dynamic programming in the design of parallel prefix networks. Journal of Functional Programming, 21:1, pp. 59–114, Cambridge University Press, 2011.
   Number of citations: 0
2. K. Claessen, N. Een, M. Sheeran, N. Sörensson, A. Voronov and K. Åkesson. SAT-Solving in Practice, with a Tutorial Example from Supervisory Control. Discrete Event Dynamic Systems, pp. 495–524, Vol. 19, issue 4, Springer, 2009. (Note: Google Scholar shows 13 citations, but I believe this to be incorrect.)
   Number of citations: 4

3. 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: 39
4. (∗) 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, pp. 349–358, May 2003.
   Number of citations: 8

2. Articles in refereed collections and conference proceedings (2003–2011)

1. E. Axelsson, Emil, K. Claessen, M. Sheeran, J. Svenningsson, Josef; D. Engdal, A. Persson. The Design and Implementation of Feldspar: an Embedded Language for Digital Signal Processing. IFL 2010, the 22nd Symposium on Implementation and Application of Functional Languages (accepted to appear in post-symposium proceedings after refereeing, to appear 2011).
   Number of citations: 0
2. Emil Axelsson, Gergely Dévai, Zoltán Horváth, Karin Keijzer, Bo Lyckegård, Anders Persson, Mary Sheeran, Josef Svenningsson and András Vajda. Feldspar: A Domain Specific Language for Digital Signal Processing algorithms. In Proc. Eighth ACM/IEEE International Conference on Formal Methods and Models for Codesign, MemoCode, IEEE Computer Society, 2010.
   Number of citations: 5
3. J. Svensson, M. Sheeran and K. Claessen. GPGPU Kernel Implementation and Refinement using Obsidian. Proc. Seventh International Workshop on Practical Aspects of High-level Parallel Programming, ICCS, Procedia, 2010.
   Number of citations: 6
4. Gergely Dévai, Máté Tejfel, Zoltán Gera, Gábor Páli, Gyula Nagy, Zoltán Horváth, Emil Axelsson, Mary Sheeran, András Vajda, Bo Lyckegård and Anders Persson. Efficient Code Generation from the High-level Domain-specific Language Feldspar for DSPs. In Proc. ODES-8: 8th Workshop on Optimizations for DSP and Embedded Systems, assoc. with IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2010.
   Number of citations: 4
5. 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*, 2008. Springer LNCS 5386. (accepted to appear in post-symposium proc. after refereeing)
   Number of citations: 7
6. K. Subramaniyan, E. Axelsson, M. Sheeran and P. Larsson-Edefors. Layout Exploration of Geometrically Accurate Arithmetic Circuits. Proceedings of IEEE International Conference of Electronics, Circuits and Systems, 2009.
   Number of citations: 1

7. 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: 7
8. 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: 3
9. 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: 2
10. 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: 14
11. E. Axelsson, M. Björk and M. Sheeran. Teaching Hardware Description and Verification. In *Proc. International Conference on Microelectronic Systems Education*, IEEE, 2005.
    Number of citations: 5
12. 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: 31
13. M. Sheeran. Hardware design and functional programming: a perfect match (invited paper). In *Proceedings 9th Brazilian Symposium on Programming Languages (SBLP05)*, 2005. (Note: All citations are of the extended Journal version, which has 39 citations.)
    Number of citations: 0
14. (∗) M. Sheeran. Generating fast multipliers using clever circuits. In *Proc. Int. Conf. on Formal Methods in Computer-Aided Design, FMCAD'04*, Springer LNCS 2312, pp. 6-20, 2004.
    Number of citations: 43
15. 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: 9
16. (∗) M. Sheeran. Finding regularity: describing and analysing circuits that are almost regular. In *Proc. Int. Conf. on Correct Hardware Design and Verification Methods, CHARME'03*, Springer LNCS 2860, 2003.
    Number of citations: 11

3. Other papers, edited proceedings, (2003–2011)

1. 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.
   Number of citations: 0

2. W. Swierstra, K. Claessen, C. Seger, M. Sheeran and E. Shriver. Chalk: a language and tool for architecture design and analysis. Workshop on Designing Correct Circuits, associated with ETAPS, 2010. (accepted based on refereeing of abstract).
   Number of citations: 0
3. 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)
   Number of citations: 1
4. 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.
5. 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
6. 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. (This is a tech. report, but it is included because it has attracted citations.)
   Number of citations: 10
7. 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

5. Freely available computer programs

1. The Feldspar language and compiler for Digital Signal Processing algorithm design is released (by Ericsson) as open source software. Sheeran's main role is in developing the necessary programming idioms, and associated tutorial materials. Release 0.4 is the current version. `http://feldspar.inf.elte.hu/feldspar/`
2. The Haskell program for prefix network design and visualisation, associated with the journal paper "Functional and dynamic programming in the design of parallel prefix networks", JFP 21:1, is freely available via a link given in the paper. `http://www.cse.chalmers.se/~ms/PPSearch/`

6. Popular scientific presentations

M. Sheeran. Tutorial presentation on Domain Specific Langauges, Ericsson Software Research Day, 2009.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2011-11863-89113-49

Dnr
2011 -

Name of applicant
Sheeran, Mary

Date of birth
590310-2266

Reg date
2011-04-13 12:52:51

Project title
A functional programming approach to hardware acceleration of algorithms

NT

Project research grant NT 13 April 2011
Project Research Grant
Computer Science
Chalmers tekniska högskola

Appendix S

NT-S

Signatures

2012	2013	2014	2015	2016	
929	938	985	995		2012-01-01 -- 2015-12-31

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 applicant 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

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

Vetenskapsrådets noteringar
Kod

2011-11863-89113-49

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se