Scalable and parameterised VLSI architecture for efficient sparse approximation in FPGAs and SoCs

F. Ren, W. Xu and D. Marković

A parameterised and scalable very large scale integration (VLSI) soft intellectual property (IP) is presented that can be implemented in programmable logic devices, such as field programmable gate arrays (FPGAs) or a system-on-chip design for efficient sparse approximation. The proposed architecture is optimised based on the orthogonal matching pursuit algorithm by both algorithm reformulation and architecture resource sharing techniques. The soft IP core supports a floating-point data format with 10 design parameters, which provides the necessary flexibility for application-specific customisation. The soft IP is evaluated on various FPGA platforms. The evaluation results show that design can achieve up to 30% higher throughput than the existing solutions while offering a larger dynamic range capability and better design flexibility.

Introduction: In recent years, the principles of sparse approximation have been widely exploited and applied to a wide range of research problems, including compressive sensing (CS), classification, data separation etc. However, due to the high computational complexity of the sparse approximation algorithms, existing software solutions based on general-purpose computing platforms are neither energy-efficient nor cost-effective for real-timing processing purposes. To enable efficient computation, several dedicated very large scale integration (VLSI) designs have been proposed [1–4]. Nonetheless, these solutions are customised towards a certain problem and lack the flexibilities for accommodating other different application specifications, such as problem size, throughput, power budget etc. Such a deficiency in flexibility has largely limited their application scope.

In this Letter, we present a flexible and reusable VLSI soft intellectual property (IP) for efficient sparse approximation. It can be implemented in programmable logic devices, such as field programmable logic gate arrays (FPGAs) or a system-on-chip (SoC) design for a variety of real-time embedded applications. The IP core supports a floating-point data format with 10 design parameters, providing the necessary flexibility for design space exploration and application-specific customisation. The proposed architecture is designed based on the orthogonal matching pursuit (OMP) algorithm [5] considering both algorithmic and architecture level optimisations.

OMP algorithm optimisation: Our previous paper has shown that the least-square (LS) solving task of the OMP algorithm plays a pivotal role in the efficiency of the implementation [4]. To improve the efficiency of the IP core, we manage to significantly reduce the complexity of the LS task through a combination of algorithm reformulation techniques.

Square-root-free Cholesky factorisation: The LS problem to be solved at iteration t of the OMP algorithm is given by $\min\|y-\boldsymbol{A}_{\Lambda_t}x(\Lambda_t)\|_2^2$, where (in the context of CS) $y\in\mathbb{R}^m$ is the measurement, Λ_t is the current active set, $\boldsymbol{A}_{\Lambda_t}\in\mathbb{R}^{m\times t}$ are the active atoms of the sampling matrix and $x(\Lambda_t)\in\mathbb{R}^t$ is the current estimation of the original k-sparse signal $x\in\mathbb{S}_k^n$. The solution to the LS problem has an analytical form and can be computed by solving the normal equation

$$\boldsymbol{\Phi}_t x_t(\Lambda_t)=\boldsymbol{A}_{\Lambda_t}^{\mathrm{T}}y \qquad (1)$$

where $\boldsymbol{\Phi}_t=\boldsymbol{A}_{\Lambda_t}^{\mathrm{T}}\boldsymbol{A}_{\Lambda_t}\in\mathbb{R}^{t\times t}$ is a positive definite matrix. Note that if QR decomposition is used to solve (1) as $\boldsymbol{A}_{\Lambda_t}=\boldsymbol{Q}_t\boldsymbol{R}_t$, a memory space of $mk+k^2/2$ words will be needed for storing both $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$. Alternatively, Cholesky factorisation is used as $\boldsymbol{\Phi}_t=\boldsymbol{L'}_t\boldsymbol{L}_t^{\mathrm{T}}$ in our design. In this case, only $k^2/2$ words are necessary for storing $\boldsymbol{L'}_t$. Given $m>2k$ [1–4], over five times memory size reduction can be achieved by choosing the proper factorisation method. Since the OMP algorithm is a heavily memory-bounded algorithm, such memory size saving can lead to significant area reduction for the overall design.

In the conventional Cholesky factorisation method, t square-root operations are involved in computing the diagonal elements of $\boldsymbol{L'}_t$. Note that an explicit implementation of this nonlinear operation is not cost-effective as it requires a large amount of logic resources and cannot be reused by other tasks. To address this design challenge, we adopt an alternative Cholesky factorisation method, which essentially takes out the square-rooted factors $\boldsymbol{D'}_t$ from both $\boldsymbol{L}'$ and $\boldsymbol{L}'^{\mathrm{T}}$ as

$$\boldsymbol{\Phi}_t=\left(\boldsymbol{L'}_t\boldsymbol{D'}^{-1}_t\right)\left(\boldsymbol{D}'_t\boldsymbol{D}'_t\right)\left(\boldsymbol{D'}^{-1}_t\boldsymbol{L}\boldsymbol{D'}^{\mathrm{T}}_t\right)=\boldsymbol{L}_t\boldsymbol{D}_t\boldsymbol{L}_t^{\mathrm{T}} \qquad (2)$$

where $\boldsymbol{L}_t\in\mathbb{R}^{t\times t}$ is a lower-triangular matrix with $\mathrm{diag}(\boldsymbol{L}_t)=\mathbf{1}$, and $\boldsymbol{D}_t\in\mathbb{R}^{t\times t}$ is a diagonal matrix that no longer requires square-root operations. This reduces the computation primitives in the IP core and simplifies the resource sharing scheme in the architecture design.

Incremental Cholesky factorisation: At iteration t of the OMP algorithm, we have $\boldsymbol{A}_{\Lambda_t}=[\boldsymbol{A}_{\Lambda_{t-1}}\ \boldsymbol{a}_\varphi]$, where ϕ is the index of the new active atom. Therefore, $\boldsymbol{\Phi}_t$ in (1) can be partitioned as

$$\boldsymbol{\Phi}_t=\begin{bmatrix}\boldsymbol{\Phi}_{t-1} & \boldsymbol{A}_{\Lambda_{t-1}}^{\mathrm{T}}\boldsymbol{a}_\varphi\\ \boldsymbol{a}_\varphi^{\mathrm{T}}\boldsymbol{A}_{\Lambda_{t-1}} & \boldsymbol{a}_\varphi^{\mathrm{T}}\boldsymbol{a}_\varphi\end{bmatrix} \qquad (3)$$

In correspondence to (3), the Cholesky factorisation matrices in (2) must have the same property that

$$\boldsymbol{L}_t\boldsymbol{D}_t\boldsymbol{L}_t^{\mathrm{T}}=\begin{bmatrix}\boldsymbol{L}_{t-1} & \mathbf{0}\\ l_{21}^{\mathrm{T}} & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{D}_{t-1} & \mathbf{0}\\ \mathbf{0}^{\mathrm{T}} & d_{22}\end{bmatrix}\begin{bmatrix}\boldsymbol{L}_{t-1} & \mathbf{0}\\ l_{21}^{\mathrm{T}} & 1\end{bmatrix}^{\mathrm{T}} \qquad (4)$$

where $l_{21}\in\mathbb{R}^{t-1}$ is a column vector and d_{22} is a scalar. Equation (4) indicates that we can utilise the previous results of the factorisation matrices to simplify the computation. Specifically, instead of recomputing $\boldsymbol{L}_t$ and $\boldsymbol{D}_t$ in every step, we can update them from $\boldsymbol{L}_{t-1}$ and $\boldsymbol{D}_{t-1}$ incrementally by adding only l_{21} and d_{22} at each iteration. The computation methods for l_{21} and d_{22} can be derived by expanding (3) and (4).

Incremental estimation update: At iteration t of the original OMP algorithm, a new estimation x_t is made by solving (1), and the residue r_t is updated as $r_t=y-\boldsymbol{A}_{\Lambda_t}x_t(\Lambda_t)$. According to (1), we can derive that $\boldsymbol{A}_{\Lambda_t}^{\mathrm{T}}r_t=\boldsymbol{A}_{\Lambda_t}^{\mathrm{T}}y-\boldsymbol{A}_{\Lambda_t}^{\mathrm{T}}\boldsymbol{A}_{\Lambda_t}x_t(\Lambda_t)=\mathbf{0}$. It indicates that the updated residue is always orthogonal to the current active atoms in the OMP algorithm. In our design, we take into account this special property to further simplify the LS task. By substituting y in (1) with $\boldsymbol{A}_{\Lambda_t}x_t(\Lambda_t)+r_t$, we can derive an incremental estimation update method as the following two steps. First, the updating direction d can be computed by solving the new normal equation

$$\boldsymbol{\Phi}_t d(\Lambda_t)=c(\Lambda_t) \qquad (5)$$

where $c(\Lambda_t)=\boldsymbol{A}_{\Lambda_t}^{\mathrm{T}}r_{t-1}$. Secondly, r_t and x_t can be updated based on their previous values and d, as $r_t=r_{t-1}-\boldsymbol{A}_{\Lambda_t}d(\Lambda_t)$ and $x_t=x_{t-1}+d$. Note that $c(\Lambda_t)$ must be a one-sparse vector, where only the last element is nonzero. Consequently, the computation for $c(\Lambda_t)$ and the subsequent forward substitution (FS) can be completely bypassed.

Overall, with a combination of the above reformulation techniques, the total computational complexity of solving a LS problem in the OMP algorithm is reduced from $O(mk^3)$ to $O(mk^2)$. Specifically, after the reformulation, the LS task only involves $t+2$ inner products, a single FS and a single backward substitution (BS) at iteration t. Therefore, the operational complexity in terms of data flow control and scheduling also gets reduced.

Architecture design: The proposed VLSI architecture is shown in Fig. 1. The vector core (VC) integrates multiple PEs in parallel, and each of them can be dynamically configured through a microcode. This enables the VC to support a selected set of vector operations. A shift register logic unit is used in the feedback path of each PE for providing the folding capability to process long vectors. The scalar core (SC) integrates a comparator, a sequential divider and two adders. Depending on the top-level data-path configuration, the SC can either post-process a selected result from the VC (e.g. inner product) or process independent data in parallel. When the two cores are connected as a processing group, more advanced operations, such as correlation sorting, FS and BS can be executed. As shown in Fig. 1, the computation cores are linked by seven data-path memories. The complex data flow of the OMP algorithm is enforced by customised local memory controllers, which are synchronised by a top-level finite-state machine. Note that for achieving high performance and resource utilisation efficiency, all the computing resources in the VC and SC are shared by all the tasks of the OMP algorithm through dynamic configuration.

The proposed architecture features great scalability. Table 1 summarises all the user-defined parameters supported in our soft IP. For circuit level optimisation, the user can specify the word-length (W_M, W_E) and the pipeline stages (S_A, S_M, S_D and S_C) of each arithmetic unit to meet different precision and performance requirements. In addition, the user can also tune the memory size and the parallelism of the PEs through the architecture parameters (M, N, K and P) for accommodating different problem size and throughput specifications.

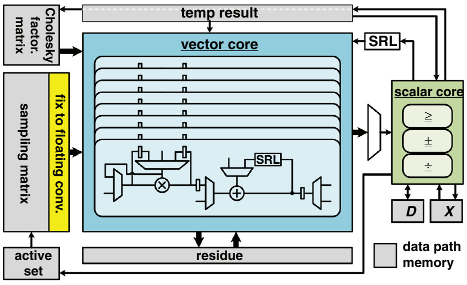

Fig. 1 Overall architecture of VLSI IP core. It supports a floating-point data format and can be configured with 10 design parameters.

Table 1: User-defined parameters in soft IP

Parameters		Descriptions
Circuits	$\{W_M, W_E\}$	Word-length of mantissa, exponent
	$\{S_A, S_M, S_D, S_C\}$	Pipe. stages of adder, mult., div. and comp.
Architectures	P	Parallelism of PEs in VC
	N	Signal dim. n
	M	Measurement dim. m
	K	Signal sparsity level k

Evaluation results: To validate the flexibility and efficiency of the soft IP, we implement multiple instances of our design with different problem sizes on different FPGA platforms. Table 2 summarises the evaluation results in comparison with recent work [1–3]. For each implementation, we take advantage of the circuit and architecture level flexibility of the soft IP to efficiently explore the mapping results of different parameter settings. Then, the one with the maximal performance and resource utilisation is selected for the final implementation. As a result of the design space exploration, our implementations are able to outperform the prior work in terms of throughput by up to 30%. In addition, as a floating-point data format is used, our design offers a much larger dynamic range capability.

Table 2: Implementation results in comparison with prior work

Designs	[1]	Our work	[2]	Our work	[3]	Our work
Platforms	Virtex-5		Virtex-6		Spartan-6	
N, M, K, P	128, 32, 5, 32		1024, 256, 36, 256		1024, 512, 64[a], 32	
P	32	32	256	256	32	32
Data format[b]	FP (32)	FLP (8, 23)	FP (25)	FLP (8, 16)	FP (30)	FLP (8, 21)
Frequency (MHz)	39	59.3	100	77.6	41.2	64.4
Slices	N/A	12 330	32 010	62 026	3525	15 769
DSP48s	N/A	64	261	256	132	98
Dec. time (μs)	24	18.5	630	581.6	21 378	17 611
Throughput[c] (K samples/s)	5333	6919	1625	1761	47.9	58.1

[a] Supported signal sparsity level is not disclosed in [3]
[b] Fixed point (FP) (word-length) and floating point (FLP) (W_M, W_E)
[c] Throughput calculated as N divided by decoding time.

Conclusion: In this Letter, we present a parameterised and scalable VLSI soft IP for implementing efficient sparse approximation in FPGAs or a SoC design. The soft IP core supports a floating-point data format with 10 design parameters, which provides the necessary flexibility for application-specific customisation. The evaluation results on various FPGA platforms confirm that by exploring the design space through parameter tuning, our design can achieve up to 30% higher throughput than the existing solutions while offering a larger dynamic range capability and better design flexibility.

9 September 2013
doi: 10.1049/el.2013.2978
One or more of the Figures in this Letter are available in colour online.

F. Ren and D. Marković (*Electrical Engineering Department, University of California, Los Angeles, CA 90095, USA*)

E-mail: renfengbo@ucla.edu

W. Xu (*Computer Science and Engineering Department, University at Buffalo, The State University of New York, Buffalo, NY 14214, USA*)

References

1 Septimus, A., *et al.*: 'Compressive sampling hardware reconstruction'. Proc. Int. Symp. Circuits and Systems (ISCAS), Paris, France, 2010, pp. 3316–3319

2 Bai, L., *et al.*: 'High-speed compressed sensing reconstruction on FPGA using OMP and AMP'. Proc. 19th Int. Conf. Electronics, Circuits and Systems (ICECS), Seville, Spain, 2012, pp. 53–56

3 Patrick, M., *et al.*: 'VLSI design of approximate message passing for signal restoration and compressive sensing', *IEEE J. Emerg. Sel. Top. Circuits Syst.*, 2012, **2**, (3), pp. 579–590

4 Ren, F., *et al.*: 'A single-precision compressive sensing signal reconstruction engine on FPGAs'. Proc. 23rd Int. Conf. Field-Programmable Logic and Applications (FPL'13), Porto, Spain, 2013

5 Tropp, J., *et al.*: 'Signal recovery from random measurements via orthogonal matching pursuit', *IEEE Trans. Inf. Theory*, 2007, **53**, (12), pp. 4655–4666