An Approximate Computing Technique for Reducing the Complexity of a Direct-Solver for Sparse Linear Systems in Real-Time Video Processing

Michael Schaffner[†‡], Frank K. Gürkaynak[†], Aljosa Smolic[‡], Hubert Kaeslin[†], Luca Benini[†]

[†] Integrated Systems Lab
ETH Zurich, Switzerland
{schaffner,kgf,kaeslin,benini}@iis.ee.ethz.ch

[‡] Disney Research Zurich
Switzerland
smolic@disneyresearch.com

ABSTRACT

Many video processing algorithms are formulated as least-squares problems that result in large, sparse linear systems. Solving such systems in real time is very demanding. This paper focuses on reducing the computational complexity of a direct Cholesky-decomposition-based solver. Our approximation scheme builds on the observation that, in well-conditioned problems, many elements in the decomposition nearly vanish. Such elements may be pruned from the dependency graph with mild accuracy degradation. Using an example from image-domain warping, we show that pruning reduces the amount of operations per solve by over 75 %, resulting in significant savings in computing time, area or energy.

Categories and Subject Descriptors

G.1.2 [**Numerical Analysis**]: Approximation
; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*Sparse, structured, and very large systems*
; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Stereo*

Keywords

Cholesky Decomposition, Hardware Accelerator, Approximate Computing, Video Processing

1. INTRODUCTION

Many image and video processing algorithms are formulated as regularized least squares problems whose solution can be found by solving a linear system of equations [13]. However, those linear systems tend to be very large - in the order of tens of thousands to millions of variables - since the unknowns are often either pixel values or pixel coordinates. Although the sparse, regular structure of such systems can be leveraged to efficiently solve them, the computational complexity is still very large. This motivates the development of custom hardware accelerators for real-time applications that operate e.g. on streaming video.

There are two main approaches that can be used to solve such linear systems [11]. Direct methods are based on matrix decomposition, whereas iterative methods calculate the solution by iteratively updating the solution vector based on e.g. a gradient descent scheme. A comparative study of hardware implementations on an FPGA by Greisen et. al. [15] shows that hardware implementations of iterative methods tend to require a large memory bandwidth and this results in a significant amount of on-chip memory - which becomes very expensive for large problems. On the other hand, direct methods such as solvers based on the Cholesky decomposition can make use of data-locality and can be efficiently realized with a small on-chip cache. Therefore, we focus on the direct Cholesky method in this paper and aim at reducing its computational complexity.

There are well established *pre-ordering* methods (such as [2, 10]) for the Cholesky decomposition that aim at reducing the number of non-zero elements in the decomposition by re-ordering of the problem matrix. While this approach can have benefits in the case where *all* elements in the decomposition are calculated by one unit sequentially (like in a CPU), it impedes the use of many parallel processing units for high performance systems due to the loss of regularity in the sparsity structure. In this paper, we pursue a different method based on approximation, which is a variant of the incomplete LU factorization with threshold (ILUT) [19] that is often used to calculate pre-conditioners for iterative methods. It has recently been shown that unconventional approximations may lead to very large savings in either computation time, energy consumption or circuit area for applications that are error tolerant to some degree (e.g. image and video processing) [3,8,17]. Bates et. al. [3,8] use the insight that *logarithmic number systems* (LNS) can be implemented very efficiently at low precisions, resulting in so called *low-precision high dynamic range* (LPHDR) arithmetic. Palem et. al. [17] propose *probabilistic pruning* - an approach where logic gates are pruned from the netlist of adders based on the statistics of the data to be processed.

In this work, we note that well-conditioned linear systems arising in video processing applications tend to contain very small elements in the *fill-in* of their Cholesky decomposition. Using example problems from *image domain warping* (IDW) [1, 16], we show that an *incomplete* factor-

DAC '14, June 01 - 05 2014, San Francisco, CA, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2730-5/14/06...$15.00.
http://dx.doi.org/10.1145/2593069.2593082

ization, where these elements are pruned from the dependency graph, still provide solutions that are accurate enough (similar as reported in [17]). This effectively reduces the number of required operations - in well-conditioned cases by more than 75 %. Incomplete factorizations essentially use a threshold in order to decide whether an element in the matrix is to be kept or not, and thus they can be conveniently integrated into existing hardware architectures.

The remainder of the paper is organized as follows: Sections 2 and 3 summarize related work and some preliminaries. The incomplete factorization technique that is used here is explained in Section 4 and numerical results are given in Section 4.2. In Section 4.3 we outline how the approach can be integrated into existing hardware architectures, and Section 5 concludes the paper.

2. RELATED WORK

The design of hardware architectures for a Cholesky decomposition based solver has been addressed in many previous studies [4–6,14,15,20,21]. Most architectures build upon an array of parallel MAC units that evaluate the abundant amount of dot products in the decomposition concurrently.

Maslennikow et. al. [14] show that rational fraction arithmetic can result in smaller operators. In [21], Sun et. al. propose a mixed-precision approach where the decomposition step is performed with low precision, followed by an iterative refinement with high precision. The idea to locally use a logarithmic number system (LNS) to improve the area and latency of divisions and square-roots is introduced in [20]. The benefits of a fused floating-point datapath is elaborated in [5]. Cho et. al. [4] as well as Yang et. al. [6] show that it can be beneficial for hardware architectures to use the modified LDL^T decomposition instead of the normal LL^T Cholesky decomposition since it does not require the calculation of square-roots and uses less divisions.

In all the work listed above, relatively small, non-sparse matrices are used, except [15] where an architecture for large, sparse linear systems in image and video processing applications is developed. As opposed to [15], our approach additionally leverages the statistics of linear systems in video processing in order to gain efficiency. Further, the method we use is orthogonal to many optimizations such as operator fusing or iterative refinement, and could therefore be combined with such optimizations as well.

3. PRELIMINARIES

3.1 Sparse Linear Systems in Image and Video Processing

Many image and video processing algorithms can be posed as a quadratic minimization problem [13,15] of the form

$$\min_{\mathbf{f}}\left(E\left(\mathbf{f}\right)\right)=\min_{\mathbf{f}}\left(E_{data}\left(\mathbf{f}\right)+E_{smooth}\left(\mathbf{f}\right)\right), \qquad (1)$$

where the data term E_{data} enforces function values at certain sampling positions, and the smoothness term E_{smooth} is a regularizer that propagates the known sample values to adjacent sampling positions. The vector $\mathbf{f}$ is holding the samples of an unknown discrete function (e.g. pixel intensities or coordinate values) defined on a two-dimensional sampling grid Λ with width w and height h. The two terms E_{data} and E_{smooth} are briefly summarized below and a more detailed description can be found in [15].

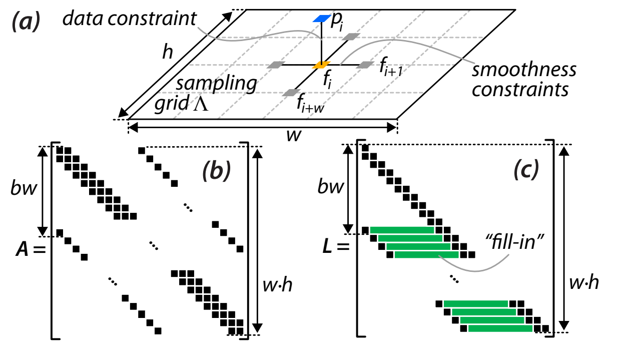

Figure 1: a) Sampling grid with unknowns f_i. b) Sparsity structure of the matrix A. c) Sparsity structure of the Cholesky factor L of A.

Let i be the *linear* index of the sampling points of the grid Λ with $i \in \mathcal{D}_\Lambda = \{1, 2, ..., w, ..., w \cdot h\}$. Figure 1 a) illustrates the relations introduced by the data and smoothness terms for one sample f_i of $\mathbf{f}$. The data term usually has the form

$$E_{data}\left(\mathbf{f}\right)=\sum_{i\in\mathcal{D}_\Lambda}\lambda_i\left(f_i-p_i\right)^2, \qquad (2)$$

where p_i are the constraint values, and the parameters λ_i are weights indicating the relative importance of the corresponding constraints. The smoothness term contains differential constraints defined among neighboring samples, and is usually given by

$$E_{smooth}\left(\mathbf{f}\right)=\sum_{i\in\mathcal{D}_\Lambda}\Big(\lambda_i^x\left(f_{i+1}-f_i-d_i^x\right)^2+ \lambda_i^y\left(f_{i+w}-f_i-d_i^y\right)^2\Big), \qquad (3)$$

where d_i^x and d_i^y are the constraint values, and λ_i^x and λ_i^y are again relative-importance-weights. The superscripts ^{x} and ^{y} indicate whether the parameter belongs to a horizontal- or vertical difference constraint. Since the energy functional $E(\mathbf{f})$ is quadratic in the elements of $\mathbf{f}$, the solution to (1) is a least-squares solution, and can be found by solving a linear equation system of the form $A\mathbf{f} = \mathbf{b}$, where A is a symmetric, quadratic and positive definite matrix. Since the constraints are defined on small, local neighborhoods on Λ, the matrix A is very sparse and only contains a main- and a few off-diagonals. Further, the number of variables in the linear system is in the order of tens of thousands to millions - depending on the resolution of Λ. The structure of A in the case of IDW will be discussed in more detail below.

3.2 Linear Systems for IDW Applications

In this paper we will use two example applications from the image processing domain: *video retargeting* [16] and *automatic stereo-to-multiview conversion* [1]. Both applications use IDW to non-linearly transform the frames of a video in content-adaptive manner. The constraints for the energy minimization problems are cues extracted from the original video frames, and the solution vector $\mathbf{f}$ represents the coordinates of the pixels in the transformed image.

Video retargeting is concerned with changing the aspect-ratio of the video frame (e.g. from 16:9 to 4:3) in a *content-adaptive* manner. Automatic stereo-to-multiview conversion uses IDW to inter- and extrapolate a given stereo 3D framepair to new virtual view positions. Key for those applications is a so called *saliency* map [12] which is used to weigh smoothness constraints in (3). This effectively moves distortions to visually unimportant regions in the image. The energy minimization problems usually also include: edge- or line constraints, a few data constraints around the image border in order to fix its position in the target image, disparity constraints in the case of stereo-video, and temporal constraints. Such temporal constraints are of the form

$$E_{data} = \sum_{i \in \mathcal{D}_\Lambda} \lambda_i \left(f_i[t] - f_i[t-1]\right)^2, \tag{4}$$

where $\mathbf{f}[t-1]$ is the already computed solution from the previous time step. This constraint is important since on one hand it ensures that there are no 'wobbling' artifacts in the transformed frames. On the other hand, it can improve the condition of the problem matrix significantly - which in turn can increase the performance of the presented pruning technique, as will be shown later. More details about the constraint formulation can be found in [1] and [16].

The resolution of current video content is predominantly 1080p (1920 pixels wide by 1080 pixels in height), resulting in two (one for each coordinate dimension) to four (when stereoscopic 3D footage is used) equation systems with nearly two million variables for each frame in the video. Solving such large systems at frame rates of up to 30 fps is computationally very demanding. Therefore, the problems are usually solved on around 10× sub-sampled grids in order to reduce the computational complexity. This results in realistic grid sizes of about 190×110, which equals to $\approx 21\,k$ variables in the minimization problem. For the application examples we consider in this paper, the problem matrix A usually has one main-diagonal and four[1] off-diagonals - out of which two are unique due to the symmetry (Figure 1 a).

3.3 Direct Solve with Cholesky Decomposition

A widely used direct-solution method for symmetric, positive definite matrices uses the Cholesky decomposition [11, 23]. This method is well known for its numerical stability and computational efficiency among direct solution methods. The standard Cholesky decomposition computes a matrix factorization of the form $A = LL^T$, where L is a lower triangular matrix. The solution to the equation system $A\mathbf{x} = \mathbf{b}$ can be conveniently found by solving $L\mathbf{y} = \mathbf{b}$ for $\mathbf{y}$ and $L^T\mathbf{f} = \mathbf{y}$ for $\mathbf{f}$ using *forward-* and *backward-substitution*.

The Cholesky decomposition is also available in a slightly modified variant which has been shown to be more attractive for hardware implementations [4, 6] thanks to fewer divisions and the absence of square-roots. The decomposition provides a factorization of the form $A = LDL^T$, where D is a diagonal matrix, and the diagonal elements of L are unity. The matrix elements are computed as

$$L_{ij} = \frac{A_{ij} - \sum_{k=1}^{j-1} D_k L_{ik} L_{jk}}{D_j} \text{ and } D_i = A_{ii} - \sum_{k=1}^{i-1} D_k L_{ik}^2, \tag{5}$$

for $i = 1, ..., n$ and $j = i+1, ..., n$. In order to obtain the solution, an additional division step has to be added between the forward and backward substitution - i.e. we first have to solve $L\tilde{\mathbf{y}} = \mathbf{b}$ for $\tilde{\mathbf{y}}$, then we perform the divisions $\mathbf{y} = D^{-1}\tilde{\mathbf{y}}$ and finally get $\mathbf{x}$ by solving $L^T\mathbf{x} = \mathbf{y}$. In this paper, we use this version of the decomposition throughout.

[1]Note that when using subsampled grids, the data constraints sometimes need to be interpolated - resulting in four additional off-diagonals right next to the outermost off-diagonals of A [15].

Computational Complexity and Sparsity of L. The decomposition step of LDL^T has a computational complexity of $\mathcal{O}\left(n^3\right)$ for $n \times n$ matrices, in general [11]. In our case, where the matrix A has the sparsity structure shown in Figure 1 b), the decomposed matrix L will be banded with bandwidth bw, and the complexity of the decomposition reduces to $\mathcal{O}\left(n \cdot bw^2\right)$ [15]. The matrix A is usually built along the smaller of the two dimensions h and w, which results in a bandwidth of $bw = \min(w, h) + 1$. The non-zero elements that appear between the main diagonal and the outermost off-diagonal during the decomposition are termed *fill-in* [15] (highlighted with green in Figure 1 c). Since the division and forward-/backward substitution steps only have complexity $\mathcal{O}(n)$ and $\mathcal{O}(n \cdot bw)$ [11], the decomposition step dominates the overall number of computations that are required to compute the solution $\mathbf{x}$.

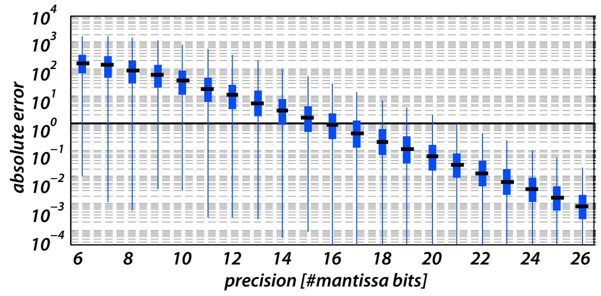

Figure 2: Precision evaluation using 10 different multiview problems with grid sizes 126×256. The figure shows boxplots of the absolute error in x. For subpixel accuracy, at least 21 bits are required here.

Arithmetic Precision and LNS. Although the Cholesky decomposition is numerically very stable, large linear systems still need to be solved using high precision in order to yield acceptable results. Figure 2 shows the results of an arithmetic precision vs. solution accuracy study for a multiview IDW problem with many data constraints. The x-axis shows the number of bits used for the mantissa of a binary coded floating point number (the amount of bits in the exponent has been fixed to 8 bits). The fewer bits are used, the smaller the resulting hardware will be. It can be seen that sub-pixel accuracy is only obtained with more than 21 bits in the floating-point format. Increased problem sizes can easily result in less well-conditioned matrices. Similarly, a reduction in the amount of data constraints can also have a negative impact on the matrix condition, as will be shown later. This can quickly mandate precisions of more than 30 mantissa bits. Unfortunately, LNS-based arithmetic cannot be used to reduce the hardware complexity here. The main operation of the Cholesky decomposition is the long scalar product for which LNS is not competitive when high precision is required [7].

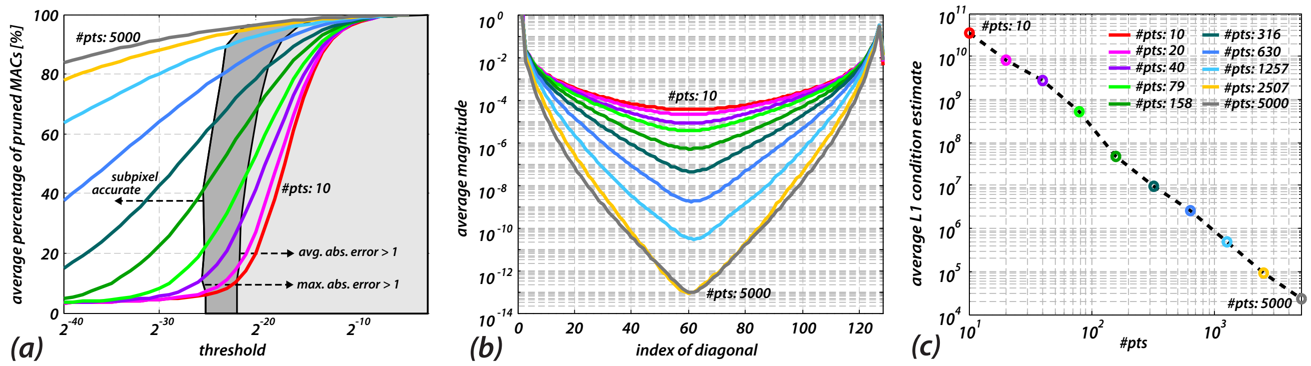

Figure 4: Evaluation of the pruning on a set of multiview problems (grid size 126×256) with an increasing amount of random data constraints. a) shows the amount of pruned MAC operations; b) shows the average magnitude of L, averaged along the diagonals; and c) shows the averaged L1 condition estimate (condest **from MATLAB) of the problem matrices. The regions where the maximum- and mean errors are greater than one are shaded in a).**

4. PRUNING OF OPERATIONS

4.1 Observations and Incomplete Factorization

An investigation of the fill-in values in decomposed problem matrices from IDW applications revealed, that many of them are very small compared to the elements on the main- and outermost off-diagonal. Consider the example in Figure 3 a), where the average magnitude on different diagonals of a decomposed multiview problem is shown. One can observe a large magnitude difference between elements located in the middle and elements on the border of the band. Therefore, we apply a pruning technique that leverages this fact by noting that the new fill-in elements in the Cholesky decomposition are calculated with inner products among already computed rows of L. The idea is to skip multiplications with very small operands, since these are unlikely to have a large contribution to the final scalar product sum - i.e. an element L_{ij} from the dependency graph is removed if its magnitude is below a certain threshold ρ. Numerically this has the same effect as

$$L_{ij} := \begin{cases} 0 & \text{if } |L_{ij}| < \rho \\ L_{ij} & \text{else} \end{cases}. \tag{6}$$

In hardware, the information whether an element L_{ij} is pruned can be directly used to skip MAC operations with a pruned operand *at runtime*. This may be implemented by using an array of valid-bits associated with the L_{ij} elements. Note that just statically skipping off-diagonals with low average magnitude is not feasible, since they still may contain important elements, as can be seen in Figure 3 b). It should be mentioned at this point that the above thresholding technique itself is not new, and belongs to the class of so called *incomplete LU factorizations with threshold* (ILUT), which are often used to calculate pre-conditioners for iterative methods [19]. The difference to our work is that we directly use the incomplete factorization to calculate a solution of the linear system, since this provides enough accuracy for the applications at hand.

4.2 Evaluation

The incomplete factorization scheme is a heuristic and depends on the statistics of the data to be processed. However, its behaviour is stable if the problem matrix is well-conditioned, and an appropriate value for ρ is used. In the following we present numerical evaluations showing the influence of the amount of data constraints and the value of ρ on the performance of the technique. Further, an evaluation using multiview and retargeting video sequences is shown. The video footage used here is taken from [18, 22].

Data Constraints and the Value of ρ. In order to investigate the effect of data constraints and the threshold ρ, we used a data set of 10 natural images to generate problem matrices with an increasing number of disparity data constraints. The disparity data was generated at random, such that the amount of constraints is exactly the same for all 10 test images (this is important since the results from the 10 natural images were averaged). The evaluations were performed with double precision in order to minimize quantisation effects[2]. In order to calculate the error, the solutions were compared against the same implementation without pruning.

[2]Note that the pruning scheme can also be combined with reduced precision arithmetic - e.g. single precision.

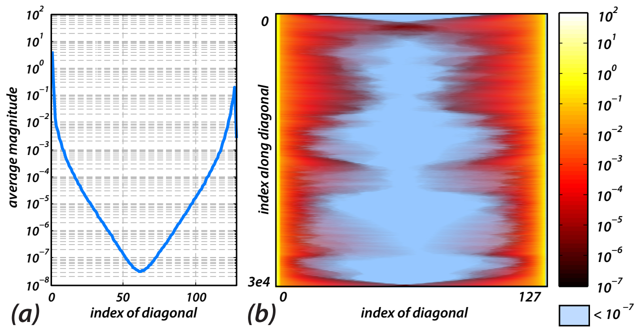

Figure 3: Magnitude of the elements of L, once averaged along the diagonals in b), and once shown in two dimensions in a). The main diagonal of L has been replaced with the values from D here. The blue elements in b) have a magnitude less than 10^{-7}.

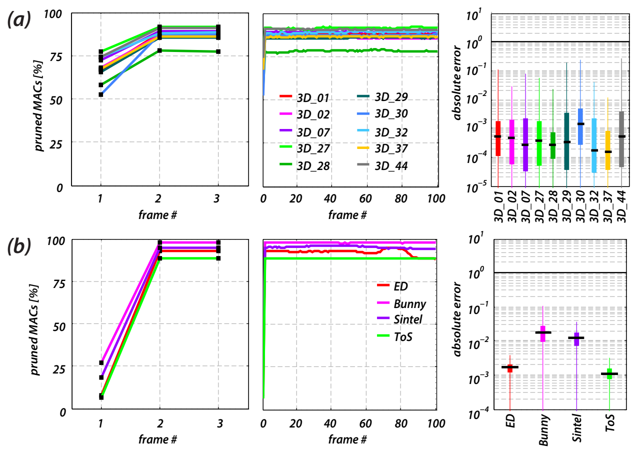
Figure 5: Evaluation on multiview- a) and retargeting video sequences b). The plots only show results for the x-problems (the corresponding y-problems tend to have better performance).

As can be seen in Figure 4 a), the more (uniformly distributed) data constraints are available, the better the performance of the incomplete factorization approach. An intuitive interpretation of this behaviour is that, in the case where only a few data constraints are available, the values of most elements in the solution vector have to be inferred by propagating the few data constraints across large parts of the equation system. This leads to tightly coupled systems with a flat curve in Figure 4 b). In the other case where many data constraints are available, we find the opposite: the data constraints have only local influence on the values in the solution vector, which results in less coupling and a v-shaped curve in Figure 4 b). This indicates that many elements in L are insignificant and can be pruned away. Since data constraints are formulated such that they are added to the diagonal of the A matrix, problems with many data constraints have the tendency to be much better conditioned, as can be seen in Figure 4 c). Typical multiview problems usually have many disparity constraints (> 1000), an on the basis of this evaluation we can see that by setting the threshold ρ to around 2^{-25}, more than $75\,\%$ of all MAC operations can be skipped, with only loosing subpixel accuracy.

Video Sequences. The second evaluation comprises 10 multiview video sequences and 4 retargeting sequences of 100 frames. Note that all problem matrices contain a temporal consistency constraint - except the ones belonging to the first frames. The pruning threshold has been set to 2^{-27} for the multiview sequences, and to 2^{-29} for the retargeting sequences. All cases were evaluated using double precision. The grid sizes are 126×256 and $\{200, 204, 270\} \times 480$ in the case of multiview and retargeting problems, respectively.

Considering the results for the first frames in Figure 5, we can see that the retargeting problems perform much worse than the multiview problems. This is because the retargeting problems only contain a few data constraints around the image borders, whereas the multiview problems contain many, well-distributed data constraints. However, the amount of pruned operations changes drastically for all subsequent frames. This is due to the temporal consistency constraint which introduces many data constraints all around the image - and which is absent in the first frame. We observe that in all evaluated sequences the use of an incomplete factorization can reduce the amount of required MAC operations by more than $75\,\%$. Note that the solution is always sub-pixel accurate, and that the pruning threshold is held constant throughout the entire sequence.

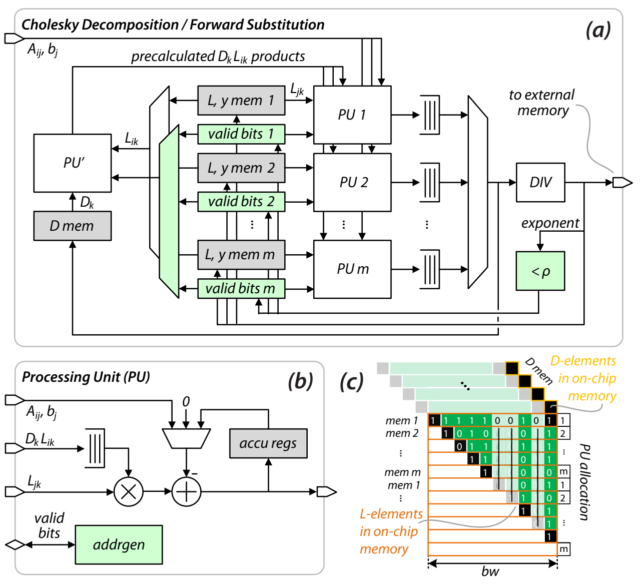
Figure 6: a) tentative hardware architecture. b) detailed view of a processing unit. c) allocation of the processing units and memories. Blocks that have to be added in order to support pruning are highlighted in green in a) and b). A possible distribution of valid bits is overlaid in c). The vertical black bars mark columns that are masked by zero $D_k L_{ik}$ products.

4.3 Integration Into Existing Hardware Architectures

The incomplete factorization technique can be conveniently integrated into hardware architectures that build upon an array of MAC units, such as presented in [6]. The important feature is that individual MAC units can independently decide whether an operation can be skipped. A possible hardware architecture for the decomposition- and forward substitution steps is outlined in Figure 6 a). The backward substitution is not elaborated further since it can be simply implemented using one MAC unit.

Since it is infeasible to store the whole L matrix on-chip due to its size, an off-chip memory is required. As shown in [15], the band-structure of L allows to cache only a small part of the decomposed matrix in an on-chip buffer. The datapath of the architecture contains an array of m processing units (PU), each of which is connected to its own RAM bank. As shown in 6 c), each PU is allocated to a subset of rows in the decomposed matrix L, and the corresponding RAM bank contains the previously calculated elements of that row. An additional processing element (PU') is responsible for the precalculation of the $D_k L_{ik}$ products. PU' is connected to all RAM banks, and to an additional RAM containing the D_k values. The $D_k L_{ik}$ are broadcasted to PU_1 - PU_m, which use them to build the scalar products with the rows they are allocated to. The structure of a PU is shown in 6 b).

Enhancements for Incomplete Factorization. Pruning can now be implemented by adding a set of valid bits to PU_1 - PU_m, and by enhancing the address generators of the processing units. Each set of valid bits has to be able to hold $bw \cdot \lceil bw/m \rceil$ bits, and - by using these bits - the address generators have to be able to determine the next valid L_{jk} addresses in the on chip memory within one cycle. Note that invalid $D_k L_{ik}$ products mask whole columns in L, as indicated with black lines in subfigure c). These can be determined by PU' during precalculation, such that PU_1 - PU_m can skip them. These enhancements complicate the address generation logic, but since the dominant circuit area is attributed to the floating point operators, the enhancements are expected to have low overhead in the order of 10 %.

Savings. In the following, we neglect the impact of any overhead introduced by the forward substitution step and by the latency of floating point operators[3]. If we have the extreme case $m = 1$, where we have one PU that evaluates all MAC operations, it is possible to leverage all pruned MAC operations - i.e. more than 75% of the cycles required to complete a decomposition can be saved in the case of the IDW examples. If m is now increased, those savings gradually get smaller, and are lower bounded by the savings obtained in the other extreme where we have $m = bw$. In this scenario, the limiting factor is the amount of cycles required to calculate the $D_k L_{ik}$ elements, which is given by the number of non-zero MAC operands. In all examples shown in this paper, less than 50% of the matrix elements are non-zero. Note that this can also translate into large savings in terms of external memory bandwidth if only non-zero elements are written to the off-chip memory.

5. CONCLUSIONS AND FUTURE WORK

We showed that, for least squares problems typically arising in video processing applications, incomplete Cholesky factorizations can be used to calculate approximate solutions with only mild impact on the accuracy.

The prerequisite for this approach is that the linear systems have to be diagonally dominant - which is often the case when many data constraints or temporal constraints are present. We showed that the approach can save more than 75% of the MAC operations in two different IDW applications, and that convenient integration into existing hardware architectures is possible.

At the moment, no analytical bounds on the expected performance can be given - which mandates numerical simulations with application specific datasets in order to assess the performance and determine the pruning threshold. However, we believe that the approach could also have benefits in other video processing applications with similar structure - especially if they contain temporal consistency constraints.

A proof-of-concept implementation of the proposed architecture is in progress, and we also plan to revisit matrix reordering techniques in this context, since they can be combined with incomplete factorizations.

[3]The overhead due to the latency of the floating point operators may be minimized by using fused operators, e.g. [9].

6. REFERENCES

[1] A. Smolic et. al. Disparity-Aware Stereo 3D Production Tools. In *CVMP*, pages 165–173, 2011.

[2] P. Amestoy, T. Davis, and I. Duff. Algorithm 837: AMD, An Approximate Minimum Degree Ordering Algorithm. *ACM TOMS*, 30(3):381–388, 2004.

[3] J. Bates. Processing With Compact Arithmetic Processing Element, Dec. 24 2010. WO Patent 2,010,148,054.

[4] H. Cho, J. Lee, and Y. Kim. Efficient Implementation of Linear System Solution Block Using LDL^T Factorization. *SoC 2008*, 03, 2008.

[5] S. Demirsoy and M. Langhammer. Cholesky Decomposition Using Fused Datapath Synthesis. In *ACM/SIGDA FPGA 2009*, pages 241–244, 2009.

[6] Y. Depeng, G. D. Peterson, and H. Li. Compressed Sensing and Cholesky Decomposition on FPGAs and GPUs . *Parallel Computing*, 38(8):421 – 437, 2012.

[7] J. Detrey and F. De Dinechin. A Tool for Unbiased Comparison between Logarithmic and Floating-point Arithmetic. *J VLSI SIG PROC SYST*, May 2007.

[8] R. S. Eaton, J. C. McBride, and J. Bates. Reliable ISR Algorithms for a Very-low-power Approximate Computer. In *SPIE DSS*, pages 871312–871312, 2013.

[9] F. De Dinechin et. al. An FPGA-Specific Approach to Floating-point Accumulation and Sum-of-products. In *ICECE Technology, 2008. FPT.*, pages 33–40, 2008.

[10] A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 1973.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 3. JHU Press, 2012.

[12] L. Itti, C. Koch, and E. Niebur. A Model of Saliency-based Visual Attention for Rapid Scene Analysis. *IEEE TPAMI*, 20(11):1254–1259, 1998.

[13] M. Lang et. al. Practical Temporal Consistency for Image-based Graphics Applications. *ACM ToG*, 2012.

[14] O. Maslennikow et. al. Parallel implementation of Cholesky LL^T-Algorithm in FPGA-based processor. In *PPAM*, pages 137–147. Springer, 2008.

[15] P. Greisen et. al. Evaluation and FPGA Implementation of Sparse Linear Solvers for Video Processing Applications. *IEEE TCSVT*, Aug. 2013.

[16] P. Krähenbühl et. al. A System For Retargeting of Streaming Video. *ACM ToG*, 28(5):1, Dec. 2009.

[17] K. Palem and A. Lingamneni. Ten Years of Building Broken Chips: The Physics and Engineering of Inexact Computing. *ACM TECS*, 12(2s), May 2013.

[18] RMIT Univ. An Uncompressed Stereoscopic 3D HD Video Library, Nov. 2013. http://www.rmit3dv.com.

[19] Y. Saad. Iterative Methods for Sparse Linear Systems Second Edition. *SIAM*, 2003.

[20] D. Sonawane and M. Sutaone. High Throughput Iterative VLSI Architecture for Cholesky Factorization Based Matrix Inversion. *IJCA*, 35(8), 2011.

[21] J. Sun, G. Peterson, and O. Storaasli. High-performance Mixed-Precision Linear Solver for FPGAs. *IEEE TC*, 57(12):1614–1623, 2008.

[22] The Xiph Open-Source Community. Test Media, Nov. 2013. http://media.xiph.org.

[23] J. H. Wilkinson. A Priori Error Analysis of Algebraic Processes. In *Intern. Congress Math*, 1968.