J Supercomput (2015) 71:3974–3990
DOI 10.1007/s11227-015-1493-x

Reconfiguration based model for matrix triangulation and hardware device creation concept

Etienne Aubin Mbe Mbock[1]

Published online: 8 August 2015

Abstract This research article based on the dynamic partial reconfiguration process gives a new orientation to the idea behind the matrix inverse computation method. The conventional way of computing matrix inverse uses the "Gauss's Methods" or any optimized variant of it. This research extends the classical method limited to the optimal "Gauss's Derivative Algorithms". The new algorithm based on the "Reconfiguration Concept" has been applied and provides matrix inverse without using "Gauss's Analysis". Instead, the computation idea will be based on the reconfigured "RLP". A wide range of $n \times n$ matrix computations have been tested for their feasibility and correctness. In this paper, we assume that such an algorithm exists that can compute the inverse matrices using the partial reconfiguration concept; we then use this idea to develop a reconfiguration-based model for matrix factorization and matrix reduction. Our reconfiguration model neither utilizes any standard algorithm to achieve the decomposition or the reduction nor any related matrix concept that factorizes matrices. However, it will use the reconfiguration matrix inverse computation to triangulate matrices. The model is described by a "Vector-Matrix Equation" reconfigured from the recursive dynamic process. Since the assumptions of our model are derived from the reconfiguration matrix inverse computations, it correctly computes the triangulated matrices. Another contribution of our work is the reduction of matrices into trivial identity matrices. Our model produces triangular matrices that avoid computational failures due to matrix inverse singularity conditions; thus,this matrix factorization management approach offers a new matrix-based computation standard.

Keywords Reconfiguration analysis · Matrix triangulation · Matrix factors · Computation · Algorithms

✉ Etienne Aubin Mbe Mbock
E.mbe_mbock@stud.uni-heidelberg.de

[1] Institut für Technische Informatik Heidelberg, Heidelberg, Germany

1 Introduction

Arguably, the most important features of algorithms and process reconfiguration are part of the process that can reconfigure; these reconfigurable parts are essential not only in case of hardware programming and construction, but also in reducing the hardware complexity of the process. The last optimization feature makes the concept highly desirable in computer engineering. Differently from the most important computer engineering concepts that will derive from natural sciences and mathematics, the partial reconfiguration idea is from computer technology and is now applied in computer algorithms [1–4]. System failure due to singularities in all computational algorithms can affect the system's performance. For hardware creation this issue is of great interest because the hardware to be created must be system failure free and must have a predictable behavior. Reconfiguration means the ability of a system to modify operations. For FPGA dynamic reconfigurability the FPGA will modify operations during runtime. Partial reconfiguration will instead modify (change) a portion of an FPGA [5,6]. This concept although useful in hardware will find its overall power in hardware near systems like process, computations and algorithms. Because the hardware has to be flexible and adaptive, most hardware designers will apply partial reconfiguration in their applications. This will optimize the hardware complexity of the system ("Area Savings") which results to power saving. For this research paper "Reconfigurability" means the system is capable to reconfigure; as a consequence the system power size and cost will reduce, we refer to [7] for more details. Some other advantages are operations functions and program parts' reuse. This corresponds in the Xilinx terminology to the hardware reuse feature. Under this concept we also mean obsolescence avoidance and application portability. In this research article hardware reconfiguration will be adapted in computations and algorithms issues. Adapting this concept to algorithm has been an interesting research topic so far. Some of the achievements concern the portation of the "Kalman Filter" on an FPGA, the extension of algorithm functionality are examples that illustrate the use of this concept in the algorithmic. In addition to previous use of this concept, applications have already been successful in research. The Kalman Filter Method has been extended to the HEP Kalman Filter; the use of the concept in space applications and robotics are a few of them, refer to [8] for more details.

The concept of reconfiguration is widely used in algorithmic and the understanding of this concept has been investigated by many researchers. The use of special tools to manage and understand reconfigurations are some of the developments in the field concerning algorithms IP-over-WDM networks study with stochastic traffic has been carried using dynamic reconfiguration identifiable structures [9–12]. The use of Hthreads computational architecture enables to bridge the gap between regular programmers and complex reconfigurable devices. The use of layers of abstraction built upon standard reconfigurable devices is possible through an operating system capable of supporting a diverse collection of computational models within a reconfigurable device, see [13] for more details. Although the use of dynamic reconfiguration and its application in science and technique is new, this concept is now essential in all computer techniques fields which in turn may generate thanks to reconfiguration new knowledge. In this article, we present the reconfiguration-based model for matrix fac-

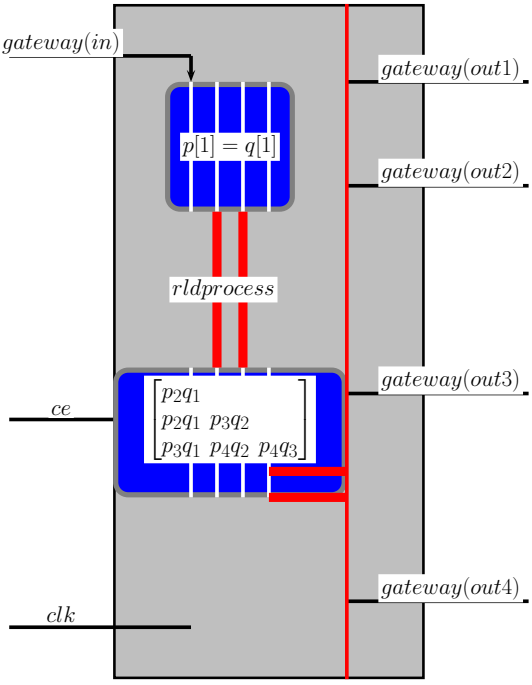

Fig. 1 Recursive linear process: *upper left* is the input state that is an integer state. The *right* side is identified by four integer states created by the recursive linear process computed by the hardware. This conceptual device is still restricted to integer values

torization. Our model combines and extends some theoretical results from the field of mathematical matrix analysis with matrix-based methods collected from several researchers relating matrix decomposition. The model we present results in reconfiguration of algorithms. More precisely the recursive process was reconfigured into a factorization method. which can be used to compute the factorization. We model the method in a "matrix-vector" equation which is a pattern that derives from the matrix inverse computation scheme, see [1]. By analyzing the matrix inverse algorithm, we obtained the analytical equation for the matrix factorization. To the best our knowledge, ours is the first reconfiguration-based model for constructing such matrix factors presented in the reconfiguration literature, as well as the first model to be translated in hardware. We demonstrate the effectiveness of our approach by comparing the results produced by our models against standard results. Figure 1 presents the recursive linear process. Table 1 summarizes the main mathematical and physical quantities used in the derivation of the proposed models and hardware construction of the "RLP" which are considered throughout this work. Tables 2 and 3 compare results produced by our model against results of standard factorization algorithms.

2 Related work and some important theoretical existing results

A few researchers have addressed the issue of technology-based partial reconfiguration [3,10,11,13–17]. Other Space-based approaches have been proposed by [8]. Essentially, they will use the reconfiguration features to enhance space applications.

Table 1 Main computer science (mathematical) and computer technology quantities considered in the development of this research paper

Symbol	Mathematical/computer science (technology) description	Physical description
q_1	Initial state of the process	Vector of length n
q_N	Final state of the process	Vector of length n
q_j	j-th state of the process	Vector of length n
VectSpace	Vector space of dimension n	$\mathbb{R}^n$
α_{ji}	States coefficients	$\mathbb{R}$
Gateway (in)	Initial state entry	Integer vector of length n
Gateway (out)	Process output	Integer vector of length n
ce	Clock enable	Hardware device
clk	Clock time	Hardware device
rldprocess (cw)	Dynamic recursive process	Hardware device
defaultclockdriver	Clock driver	Hardware device
xlpersistentdff	Device for output enable	Hardware device
R, A	Variable matrices of size $n \times n$	Real value matrices $(n \times n)$
R_{ij}, A_{ij}	Variable matrix entries	Real matrices entries $(n \times n)$
$[ij]$ $[klp]$	Integer index values	$i, j, k, l, p \in \mathbb{N}$

This is an example of a table containing the vocabulary that appears in graphics and equations, mathematical and computer science (technology) notions. This is an example of a table containing index variables, symbols, and physical definition of the notions contained in the graphics and equations

Table 2 Comparative table of some factorization methods and application in solving equations

Decomposition method	Computation elapse time (s)	Equations solving elapse time (s)
$(R\|V)$|Factorization	0.008206	0.000061
(LDU)|Factorization	0.005115	0.000587
(LU)|Factorization	0.143629	0.114312
(Cholesky)|Factorization	0.007862	0.159056
Jacobi|Iterative|Method	–	0.002752
Gauss|Seidel|Iterative|Method	–	0.005418
SOR|Iterative|Method	–	0.017114
Iterative|Refinement|Method	–	0.015308

These are selected factorization algorithms. The last four methods of the table will not factorize; they are used for application's purpose

Additionally, investigations the improvement of resource utilization have been conducted. The Hthreads techniques proposed by [18] support computational models within a reconfigurable device. These research activities make the concept of Dynamic partial reconfiguration very attractive. The discovery of the reconfiguration-based matrix computations is a significant contribution when it comes to huge matrix computations. Our research on partial reconfiguration addresses the following recently computational main discoveries:

Table 3 Comparative table of some factorization methods and application in solving equations

Decomposition method	Decomposition capacity (%)	Average complexity
$(R\|V)$|Factorization	100	$\Theta(n^2)$
(LDU)|Factorization	50	$\Theta(n^2)$
(LU)|Factorization	50	$\Theta(n^2)$
(Cholesky)|Factorization	50	$\Theta(n^2)$
Jacobi|Iterative|Method	0	$\Theta(n^2)$
Gauss|Seidel|Iterative|Method	0	$\Theta(n^2)$
SOR|Iterative|Method	0	$\Theta(n^2)$
Iterative|Refinement|Method	0	$\Theta(n^2)$

These are selected factorization algorithms. The last four methods of the table will not factorize; they are used for application's purpose

1. Reconfigurable recursive linear process.
2. Reconfigurable matrix inverse computation.

These reconfiguration-based methods will still have consequences on hardware creation and algorithms portion on FPGA. Some algorithmic techniques such as the "Kalman Filter Method" and "Matrix Inverse Computations" generate new functionality. The n dimensional Vector-Matrix model that we address in this research article introduces a product, which computes the matrix to be triangulated. This matrix will depend on the existing per reconfiguration computed upper matrices (lower matrices). The last approach proposed for which a hardware creation has been proposed will be summarized in the following formula:

$$V_n^{(k-1)} - V_n^{(0)} + \sum_{i=1}^{k-1} R_{in} \cdot V_i^{(i)} = 0 \tag{1}$$

An expansion provides the entries of the seeking matrices. One of the matrices provided by the Matlab simulator is R_{ij} with $\{i = 3, j = 4\}$ using the Matlab matrix simulator. Some of the computational conveniences of this approach concerns the matrix singularity test. In this model we will avoid computational perturbations. This avoidance is a performance indicator that guarantees the system reliability. Another convenience concerns the hardware creation as a consequence of the above idea. Since the effective hardware device creation has been the objective of our research investigations. This main goal still have to be realized. This makes this reconfiguration-based analysis worth nothing, however, that our goal was to achieve a complete computational environment in preparation to the hardware realization

$$R_{34} = \left(\frac{-A_{13} \cdot A_{14} \cdot \left(1 + A_{12}^2\right) - A_{12}^2 \cdot A_{13} \cdot A_{14} + A_{23} \cdot A_{12} \cdot A_{14}}{\left(1 + A_{12}^2\right) \cdot \left\| V_3^2 \right\|} + \frac{A_{13} \cdot A_{12} \cdot A_{24} - A_{23} \cdot A_{24}}{\left(1 + A_{12}^2\right) \cdot \left\| V_3^2 \right\|} + \frac{A_{34} \cdot \left(1 + A_{12}^2\right)}{\left(1 + A_{12}^2\right) \cdot \left\| V_3^2 \right\|} \right) \tag{2}$$

Equation (2) is a calculation of the entry [3 4] of the matrix R. All existing parameters of this equation are known and are singularity free. This advantage is robust since the calculated expression has no condition needed to be satisfied for the computations. This particular point will be addressed in the coming section. Additionally, this last feature will make hardware creation very comfortable. Since the objective of our analysis is to construct hardware we provide in Fig. 1 the conception of such hardware. Figure 1 illustrates the recursive linear process.

3 Brief overview of the triangulation method

The trivial triangulation can be deduced from the reconfigurable matrix inverse computation method. For the trivial triangulation the following matrices and vectors will be defined. These matrices are expansions of Eq. (1). Any $n \times n$ matrix can be considered to achieve this reduction:

$$\tilde{R}|\tilde{V} = \begin{pmatrix} R|V_{1,1}^{(1)} & R|V_{1,2}^{(2)} & R|V_{1,3}^{(3)} & \cdots & R|V_{1,k-1}^{(k-1)} \\ 0 & \ddots & & & \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \\ & & & 0 & R|V_{k-1,k-1}^{(k-1)} \end{pmatrix} \tag{3}$$

$$R|V_k = \begin{bmatrix} \tilde{R}|\tilde{V}_k \\ R|V_{n,k}^{(k)} \end{bmatrix} \tag{4}$$

The submatrices and vectors $\tilde{V}$; $\tilde{R}$; $\tilde{V}_k$; $\tilde{R}_k$ must be available. The following submatrix products are to be computed:

$$\tilde{V} \cdot \tilde{R}; \quad \tilde{V}_k \cdot \tilde{R}_{n,k}; \quad V_{n,k} \cdot R_{n,k}$$

The first and last product will produce identity matrix; the second product will be zero and performs the trivial reconfiguration triangulation. The non trivial triangulation will be made in n stages. The representation of this stages is summarized with the following sequence in $S = S^n S^{(n-1)} \cdots S^{(3)} S^{(2)} S^{(1)}$. The process will terminate at stage n. The following will provide the explicit description of the stages $S^{(p)}$, $p = 1, 2, \ldots, n$:

$$j = L$$
$$\sum_{j=i}^{n} R_{ij} V_{kL} \quad k = 1 \quad i = 1$$
$$j = L$$
$$\sum_{j=i}^{n} R_{ij} V_{kL} \quad k = 1 \quad i = 2$$

$$
\begin{array}{l}
\cdots\cdots\cdots \\
j = L \\
\sum_{j=i}^{n} R_{ij} V_{kL} \quad k = 1 \quad i = n-1 \\
j = L \\
\sum_{j=i}^{n} R_{ij} V_{kL} \quad k = 1 \quad i = n
\end{array}
$$

The sums in the left side of this table reconstruct the first column entries of the matrix that is to be factorized. The index i will be set to $1, 2, 3, \ldots n$ and the index k is set to 1. The second table will describe the second column entries of the decomposed matrix. This table represents $S^{(2)}$. The index l will be set to 2 and the index i will take the integer values $1, 2, 3, \ldots n$.

$$
\begin{array}{l}
j = L \\
\sum_{j=i+1}^{n} R_{ij} V_{kL} \quad L = 2 \quad i = 1 \\
j = k \\
\sum_{j=i}^{n} R_{ij} V_{kL} \quad l = 2 \quad i = 2 \\
\cdots\cdots\cdots\cdots\cdots \\
j = L \\
\sum_{j=i}^{n} R_{ij} V_{kL} \quad L = 2 \quad i = n-1 \\
j = L \\
\sum_{j=i}^{n} R_{ij} V_{kL} \quad l = 2 \quad i = n
\end{array}
$$

For the third table we set $k = L = n$ and let $i = 1, 2, \ldots n$:

$$
\begin{array}{l}
j = k \\
\sum_{j=n}^{n} R_{ij} V_{kL} \quad L = k = n \quad i = 1 \\
j = k \\
\sum_{j=i}^{n} R_{ij} V_{kL} \quad L = k = n \quad i = 2 \\
\cdots\cdots\cdots\cdots\cdots
\end{array}
$$

$$
\begin{aligned}
&j = L \\
&\sum_{j=i}^{n} R_{ij} V_{kL} \quad L = k = n \quad i = n-1 \\
&j = L \\
&\sum_{j=i}^{n} R_{ij} V_{kL} \quad L = k = n \quad i = n
\end{aligned}
$$

This analysis takes advantage of the matrix inverse computations. The starting point is the partial reconfiguration of algorithms. For this research article, we reconfigure the linear recursive process. There are significant advantages using this method. The most important advantage is the complexity time that is better than any LU decomposition algorithm to the best of our knowledge. The Factorization is unique and no permutation of the constructed matrix is needed. Some computations related to Gauss will not be necessary except for computational check. This method will be very important in all matrix-based issues, especially in image processing when it comes to represent images as matrices. Graph theory will also take advantage with this method in computational mechanic by solving linear systems of equations. In code theory we can classify all triangulation (Factorization). Solving linear system of equation will be greatly improved. This method will be better than all existing iterative methods. Large and small matrices will be equally handled. All $n \times n$ matrices manipulation will be reasonably computed with the triangulation method presented in this paper. More precision will be provided in the coming sections with a comparatives speeds table that analyses the execution time of various decomposition algorithms.

4 Existing models for matrix triangulation and discussion

The computational matrix-based literature describes several models built around two major algorithms in matrix decomposition:

1. The LU-Decomposition algorithm is an explicit method designed to factorize any non singular $n \times n$ matrix A. This method performs by considering permutations. Although this method is widely used for solving mechanical problems, it presents some disadvantages. The principal submatrices of any matrix A must be all non-singular. This method will be practicable under the assumption that the matrix A is non singular. The analysis of this method is based on the following elimination method:

$$
\begin{aligned}
&\text{Do for each } j \text{ such that } k+1 \leq j \leq n \\
&\text{row}(j) - m_{jk}\,\text{row}(k) \longrightarrow \text{row}(j)
\end{aligned}
$$

   This method will not work unless the m_{jk} are all well defined. By varying j a sequence of lower triangular matrices will be formed and its product is the factor matrix L. The upper triangular factor will be obtained by taking the inverse of the matrix product.

2. Cholesky factorization is designed to compute factorization for matrices that are positive definite. This method will not decompose matrices with entries that are zero in the main diagonal of the matrix. Due to the following initialisation of the Cholesky decomposition method,

$$L_{(kk)} \longleftarrow \sqrt{(a_{kk})} \tag{5}$$

complex factorization will be allowed instead of real-valued decomposition.
3. The computational analysis of the two provided methods is not new; there are optimized variants of this algorithm in research. All these variant are basically subject to singularities. To avoid that issue a partial reconfiguration-based model for factorization is developed in the coming sections. There are few recent achievements on partial reconfiguration of algorithms, all addressing algorithm optimization and algorithm creations. The $[Q, R]$-decomposition method can now be achieved with partial reconfiguration. Some analyses have been made on algorithms that can reconfigure with applications in real vectors coding and applied robotic. Partial reconfiguration of algorithms will extend algorithm functionality and will allow a systematic hardware creation. The previously cited objective of partial reconfiguration is significant in computer sciences and the recent matrix inverse computations and the factorization-based methods attest this point. For the coming matrix factorization model description we assume the following facts:
   (a) All constructed matrices are considered as linear mappings of dimension $n \geq 1$. The vectors in this research article are all elements of the vector spaces $\mathbb{R}^n$.
   (b) The norm of a column vector A_j denoted $\|A_j\|$ is the root of square scalar of A_j.
   (c) The matrix with all entries equal to zero is the zero matrix and the matrix which has all entries that are not diagonal equal to zero and all diagonal entries equal to 1 is the identity matrix.
   (d) The recursive linear process is given and the partial reconfiguration matrix inverse algorithm as well.

5 Proposed factorization-based model theorem

The switch description given through [Eq. (1)] is based on the dynamic recursive process of reconfiguration. The factorization in triangular matrix factors is given by the following lighting theorem that gives a condition under which the LDU Factorization Method and the "Reconfiguration Method" factorization result in the same $n \times n$ matrix:

Theorem 1 *The LDU method and the Reconfiguration Decomposition factorize the same matrix $n \times n$ matrix H if the condition*

$$\sum_{\substack{k \leq j \leq n \\ 1 \leq i \leq n \\ 1 \leq k \leq n}} a_{kj} b_{ji} = \sum_{\substack{1 \leq j \leq k \\ 1 \leq i \leq n \\ 1 \leq k \leq n}} \tilde{a}_{kj} \tilde{b}_{ji} \tag{6}$$

is satisfied where the coefficients $\tilde{a}|a_{i,j}$ $\tilde{b}|b_{i,j}$ *are the factorization entries of the matrix* H.

The proof idea of this theorem is based on the fact that, given any $n \times n$ matrix using the "RLP" can reconfigure the product of triangular matrices via Eq. (1). The extensions of this equation lead to Eq. (3). At stage $k = 2$, Matrix H can be specified in the following matrix:

$$\begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{12}b_{22} \\ a_{22}b_{21} & a_{22}b_{22} \end{bmatrix} = \begin{bmatrix} R_{11}V_{11} + R_{12}V_{21} & R_{12}V_{22} \\ R_{22}V_{21} & R_{22}V_{22} \end{bmatrix}.$$

On the other side, the LDU provides the product of a lower triangular with an upper triangular matrix. The product is given as follows:

$$\begin{bmatrix} \tilde{a}_{11} & 0 \\ \tilde{a}_{21} & \tilde{a}_{22} \end{bmatrix} \begin{bmatrix} \tilde{b}_{11} & \tilde{b}_{21} \\ \tilde{b}_{21} & 0 \end{bmatrix} = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{11}\tilde{b}_{12} \\ \tilde{a}_{21}\tilde{b}_{11} & \tilde{a}_{21}\tilde{b}_{12} + \tilde{a}_{22}\tilde{b}_{22} \end{bmatrix}.$$

This completes the proof idea at state $p = 2$ of Sect. 1 for generalization of Stage 2. We suppose that the H provided by the LDU method is made of n column vectors H_l with $l \in \{1, 2, 3, \ldots, n\}$. H is the product "LDU" and the matrix factors are at first lower and then upper matrix factors; this leads to setting $m = k$, meaning that

$$\left[\begin{bmatrix} \sum_{j=1}^{m=k} \tilde{a}_{kj}\tilde{b}_{ji} \\ \text{Set} \quad i = 1 \\ k = 1, 2, 3, \ldots \end{bmatrix} \cdots\cdots \begin{bmatrix} \sum_{j=1}^{m=k} \tilde{a}_{kj}\tilde{b}_{ji} \\ \text{Set} \quad i = n \\ k = 1, 2, 3, \ldots \end{bmatrix} \right].$$

These are all Matrix H column vectors, and they complete the proof. This method of decomposition is useful for matrices that are near singular and thus provides better results. Below we provide a table that will compare the decomposition capacity and the equation solving accuracy.

This method of decomposition will be useful for matrices that are near singular and will thus provide better results. Bellow we provide a table that will compare the decomposition capacity and the equation solving accuracy.

Table 3 compares methods that factorize and solve numerical matrix-based problems under some given conditions. The matrix we consider is of size 50×50. The measurements of the "Elapsed Time" were made after computing the "Reconfiguration $R|V$ factorization" and reconstructing in the meantime factorized matrix. The overall decomposition capacity is zero means that these algorithms will not decompose matrices. For (Cholesky|LU|LDU)-factorization, the decomposition depends upon some singularity conditions. If the matrix will have a huge condition number, this will poor the factorization. If the matrix is singular the decomposition will be impossible. These two cases reduce the factorization capability of the cited algorithms to 50 %. The decomposition capability of the proposed model in this research article is 100 %. This means the computation of the factors will not depend on the matrix singularity test. Instead will depend on the recursive linear process. This process allows $n \times n$ matrices of any kind. Although calculation and equations solving execution times vary, the theoretical complexity of factorization algorithms remains n^2 in average see [1] for more

details. We remark that such variations are less intensive if we consider the "Iterative Refinement Method, SOR Iterative Method, the Gauss Seidel Method and the Jacobi Iterative Method". Compared to all other listed methods the $R|V$-factorization has the smallest equation solving elapsed time. For these measurements we set the tolerance on 25 % and the number of iteration on 10. We observe that the factorization of any $n \times n$ lower triangular matrix will give two identity matrix factors. A condition so that the product factors yield the identity is given with the following equation system of $n \times n$ matrix equations:

$$I = \begin{cases} R \cdot V & \text{if the "LRP" is applied} \\ R \cdot V^t & \text{if factorization is identity.} \end{cases} \tag{7}$$

The reconfiguration inverse matrix computation computes the first equation, whereas the second equation is the product of the triangulation factors. We observe that the transposed matrix V^t and the matrix V are equal, creating a diagonal matrix. Since the matrix R is not zero, this observation is valid and dependent on the matrix input in the "RLP". If the factors are all identity factors, then the input matrix of the "LRP" will be lower triangular matrix. One of the features of this model is that whatever input matrix we choose, the method will always provide decomposition factors. The theoretical aspect of the process, fortunately, can make a classification of matrices according to the "LRP" response. Under symmetric conditions, the factorization process is comfortable. This means that the factor matrix will be recursively defined according to the sequence scheme defined as

$$F_n = \begin{pmatrix} & & V_{1n} \\ \boxed{F_{n-1}} & & V_{2n} \\ & & \\ 0 & & V_{nn} \end{pmatrix}. \tag{8}$$

The factorization depends on the given matrix. If the corresponding matrix is an upper triangular matrix the reconfigured factors will be the same as the non triangular input matrix. The input of the zero matrix in the reconfiguration process does not result in computation failure. Instead, the process always results in a factorization. Under some symmetrical conditions, the computation of the factorization is straightforward. The process being recursive, some matrix factors are recursively calculated. Since this is not always the case, calculations of this method will be huge. At stage 6, for example, all calculations will be impracticable. This point can address the numerical analysis adequately. In Fig. 2, we perform calculations up to stage 4, whereas the computed matrices P and R are not feasible.

5.1 Improvement of method calculation point of view and validation of model

Matrices computed by Eq. (1) will be practically computed if the stage of the calculations is less than 5. The equation is used for some matrices that are symmetric

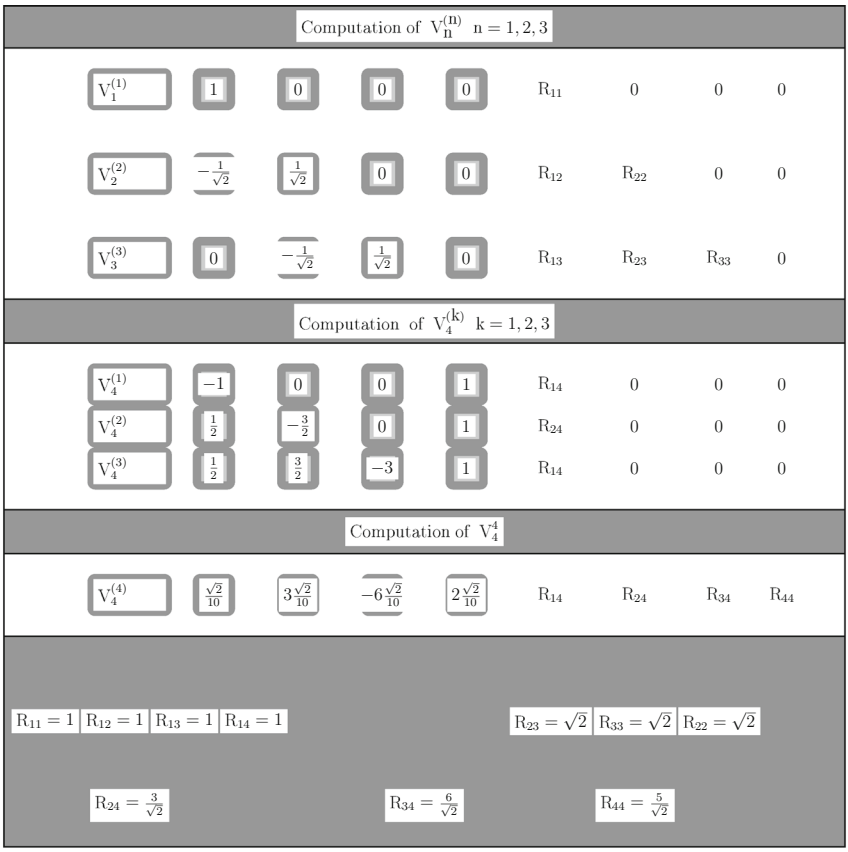

Fig. 2 Factorization calculation and computations: from table one to four we present hand calculation. Under these tables formulas are given for calculation. At the *bottom* of the figure we provide a matrix and one of its computed factors according to Eq. (1)

and recursively defined. If such matrices are restricted to their upper triangular form, Eq. (8) still applies. Equation (8) cannot be used to calculate the triangulation of all given symmetric matrices. Considering, for example, the following symmetric matrix with integer entries:

$$\begin{bmatrix} 1 & 2\,3\cdots\cdots & n \\ \vdots & & 1 \\ & \vdots & \vdots \\ \vdots & & \\ n-1 & 1 \quad \ddots & \vdots \\ n & 1\,2\cdots\cdots & n-1 \end{bmatrix}.$$

Such a symmetric matrix is not appropriate for the use of Eq. (8). In general, one cannot utilize Eq. (8) to factorize matrices per reconfiguration. In our model, however, this is possible for some simulated matrices.

6 Quantitative results and hardware construction concept

We have made some measurements on real-valued and integer-valued matrices, while the speed of the small size matrices gives the same cpu value; this remark does not apply to higher dimensional matrices. In Fig. 3, it appears that the time for the new factorization is factor 3 better than the usual lu factorization time. This is due to the reconfigurable algorithm. Our model gives exact triangulations. This is generally due to the implicit computation of the matrix to be triangulated.

In Sect. 3 we present the "RLP" in hardware with restriction to integer input values. In this section, we give details of this process in hardware. The process is made of three blocks:

1. The default-clock-driver block.
2. The xlpersistentdff block.
3. The the rdprocess block.

The expansion of the default-clock-driver block presents the following inputs $(sysce, syscl, sysclr)$. These are driven in the xlclockdriver part to output the $(ce, ce\text{-}logic, clk, clr)$. The ce and the clk are connected to the rldprocess block together with the input state of the theoretical"LRP". The xlpersistentdff block is not expandable and thus ensures that the constructed hardware functions properly. The

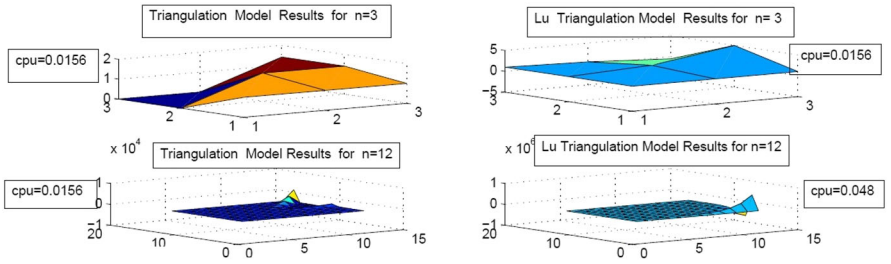

Fig. 3 CPU Comparative measurements for $n = 3, 12$. For $n = 3$ the CPU times are equal for our model whereas and for $n = 12$ the CPU times are different for the LU triangulation model

rdprocess block expansion presents the rdprocess part connected to the two previous inputs/outputs. The output of rdprocess is made of four entities ($addsub1$, $addsub2$) and ($scale1$, $scale2$). These will be transformed into the output of the system according to the following figure that only addresses the "rdprocess" block:

This construction is limited to integer values. Practically, we input the hardware with a state and the hardware creates the four output values as output states that are integers. For more details about the state outputs refer to appendix. This hardware creation although capable of generating states of any linear recursive process presents some limitations. Real-valued inputs are not accepted by this device, which is a particular point of our investigation. The second point is the construction of the hardware as a real device. The real device will be capable of using reconfiguration to solve problems related to the previous analysis. The performed qualitative comparisons in Table 2 and the results produced by our model will be useful in the following concrete applied science:

- Mathematical computational reconfiguration (Kalman Filter Method) [19]
- Space application computations using reconfigured matrices (System Failure Avoidance).
- Mathematical Programming (matrix-based and linear systems solving per reconfiguration).
- Optimization (Functionality Extension due to Reconfiguration) [19].

This concept, although addressing applied computational applied sciences and engineering fields, has some theoretical advantages. The concept gives a new vision in of coding theory. Matrix classification and new matrix representations are one of the advantages stemming from this development. The reconfiguration analysis addressing operations quantification in algorithms and computations are some concrete fields that will be of great interest to our research.

6.1 Experimentation

This research was made possible via the following software and programs:

1. Matlab Programming Environment (Mathworks R2010).
2. C/C++ Programming Environment.
3. Matlab Files (Algorithms in Matlab, Power of Pseudocodes [20]).
4. Xilinx Ise Design Suite 12.1.
5. System Generator.
6. FPGA Spartan Board Starter.
7. Latex Software.

We used Matlab R2010 because this version of Matworks is compatible with "System Generator" as part of the ISE Design Suite from Xilinx. The process was described first using Simulink, and using "System Generator" we created Figs. 1 and 4. Figure 2 is a table created to report hand calculations. This table was devised using Latex Software that creates frames for hand-calculated result reports. One should note that the computed results produced by our model closely approximate the theoretically expected results as presented in Fig. 2. This illustrates the effectiveness of our model.

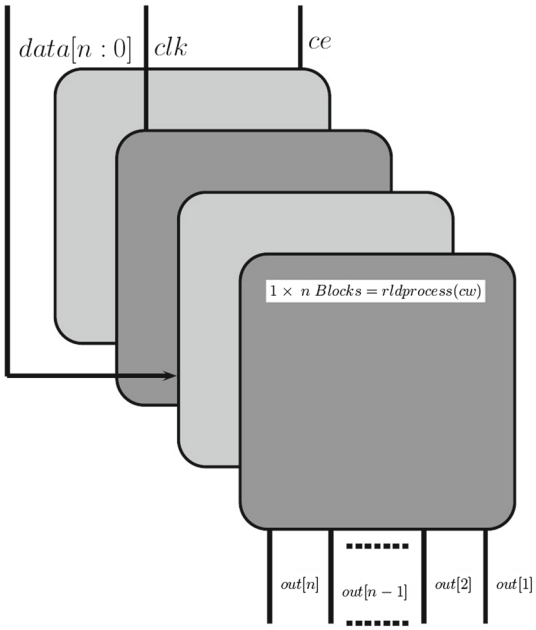

Fig. 4 Block "rdprocess" expansion. "Default-Clock-Driver" Block and "Xlpersistentdff" Blocks have been omitted since the process is valid without theses Blocks, that is, we still obtain integer results with or without the Coted Blocks

7 Discussion

We have implemented the proposed model and used it to realize factorization. The resulting decompositions are very convincing and an elapsed time achieved in Equation solving is far better than the existing standard (see Table 2). We have compared our results of factorization with the usual factorization methods. The results produced by our models are in qualitative agreement with the standard ones. In addition to this qualitative agreement, the permutation matrices will be avoided, making the factorization more precise. For this model the computation of the decomposition is given provided any $n \times n$ matrix is given. This represents an implicit matrix factorization; the ill condition problem is thus avoided. Neither the pivoting strategy nor matrix condition number has been considered in our model analysis. From a computational point of view, our model is fast and since the computation of the determinant of lower matrix is always set to 1 for the LU factorization method, our model does not predict such a lower matrix determinant. Conversely, this inconvenience is minimal as far as the overall computational complexity of the factorization is concerned. The standard methods detect ill-conditioned problems or near-singular matrices. This is not the case in reconfigurable matrix factorization algorithms. In order to demonstrate the potential use of the proposed models in computer engineering, construction of the linear recursive process has been realized on the foundational reconfiguration factorization. The construction still has to be extended to render all factorization factors. The constructed hardware provides integer states. This construction is new and can be extended to solve for real inputs, real matrix factors. This is one of the goals that we are now tracking, and the analysis we still carry based on reconfiguring of algorithm will be sufficient

for the device construction. From the concept standpoint, the use of reconfiguration influences all matrix-based computations. These aspects are not currently taken into account by many algorithm designers. The model presented contains reliable data. Matrices of any size can be handled equally. The fact that there are no conditions for implementation makes this method original not from a reconfiguration of algorithm point of view but from computations that may block a running system. Moreover, since our model covers many matrix-based computations, it can be adapted to many control theory applications, which opens an interesting avenue for future work. No standard matrix concept has been used to develop this model. For standard concepts refer to the following [21,22]. The presentations and computational point of this model could be extended too. The Gauss analysis together with LU factorization principles have not been employed to develop this model. Instead, the reconfiguration principles currently addressed in [19] have been applied. The version of the model as a technique is derived from the article [1]. Furthermore, no concept of determinants has been used. Thus, at overall angles, in addition to our new analysis, the resulting hardware construction would perceive the future hardware construction based on reconfiguration analysis. For the evaluation of our model, we used existing computational methods and special matrices. Thanks to the reconfiguration conditions, creating some interesting outcomes, such as the "Bug-free hardware effect". We use reconfiguration and the linear recursive process for our model. To the best of our knowledge, no model in the literature takes these factors into account. This is probably due to the scientific principle: concept first, then hardware. Our approach is hardware first, then concept. Our model is also original in the sense that it uses the concept of reconfiguration (hardware) to solve a computational problem. As all conditions of our models were derived from reconfiguration and the linear recursive process, it implicitly provides the factorization of any matrix. The model can be considered as a pattern since the factorization computations are exactly produced. We have validated our models through comparisons of our results against standard matrix-related algorithms (see [21,22]). Concerning our model description, Fig. 2 presents the hand calculation and the computations of our model. The quality of these presentations and computations qualitatively matched the actual standard one. To the best of our knowledge, ours is the first reconfiguration-based model for matrix factorization presented in the reconfiguration of system literature. It is also the first practical model for such factors computation to be realized in hardware. Our reconfiguration-based model for matrix-based computations is also the first model of its kind in the reconfiguration literature.

8 Conclusion

We have presented a new model for matrix factor computations. Our reconfiguration-based model combines and extends the standard existing matrix factorization algorithms presented in most matrix literature. Our results and test are basically from engineering and mathematical computing. The resulting model is expressed in terms of a "Matrix-Vector" equation that describes all the matrix factorization coefficients. This equation is general and computes factorization factors regardless of the input matrix.

References

1. Mbock EAM (2014) Algorithm based partial reconfiguration with application on matrix inverse computations. In: Transactions on engineering technologies. Springer, Berlin (ISBN:978-94-017-9114-4)
2. Vaidyanathan, Trahan (2004) Dynamic reconfiguration: architectures and algorithms. In: Series in computer sciences. Springer, Berlin
3. Arabnia HR, Oliver MA (1987) Arbitrary rotation of raster images with SIMD Machine architectures. Int J Eurograph Assoc (Comput Graph Forum) 6(1):3–12
4. Bhandarkar SM, Arabnia HR, Smith JW (1995) A reconfigurable architecture for image processing and computer vision. Int J Pattern Recogn Artif Intell (Spec Issue VLSI Algorithms Archit Comput Vis Image Process Pattern Recogn AI) 9(2):201–229
5. Xilinx (2010) Xilinx early access partial reconfiguration with planahead 9.2 users guide. http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_1/ug702.pdf. Accessed 03 May 2010
6. Xilinx, Inc (2003) Two flow of partial reconfiguration: module-based and difference-based technical report. http://www.xilinx.com/support/documentation/application_notes/xapp290.pdf. Accessed 03 Dec 2007
7. Kao C (2005) Benefits of partial reconfiguration. Xcell J 55:65–68
8. Osterloh B, Michalik H, Habinc SA, Fiethe B (2009) Dynamic partial reconfiguration in space applications. In: Conference publication, pp 336–343
9. Brzezinski A, Modiano E (2005) Dynamic reconfiguration and routing algorithms for IP-over-WDM networks with stochastic traffic. In: Proceedings of IEEE infocom
10. Arabnia HR, Bhandarkar SM (1996) Parallel stereocorrelation on a reconfigurable multi-ring network. J Supercomput (Springer Publ) 10(3):243–270
11. Arabnia HR, Oliver MA (1987) A transputer network for the arbitrary rotation of digitised images. Comput J 30(5):425–433
12. Bhandarkar SM, Arabnia HR (1995) The hough transform on a reconfigurable multi-ring network. J Parallel Distrib Comput 24(1):107–114
13. Peck W, Anderson E, Agron J, Stevens J, Baijot F, and Andrews D (2006) hthreads: a computational model for reconfigurable devices. In: Proc. of international conference on field programmable logic and applications, pp 885–888
14. Donato A, Ferrandi F, Santambrogio M, Sciuto D (2005) Operating system support for dynamically reconfigurable soc architecture. In: IEEE international soc conference, pp 233–238
15. Lübber E, Platzner M (2009) Reconos: multithreaded programming for reconfigurable computers. In: ACM Transactions Embedded Comput Syst 9(1)
16. Williams JA, Bergmann NW (2004) Embedded linux as a platform for dynamically self-reconfiguring systems-on-chip. In: Proc. of international conference on engineering of reconfigurable systems and algorithms, pp 163–169
17. Bhandarkar SM, Arabnia HR, Smith JW (1995) A reconfigurable architecture for image processing and computer vision. Int J Pattern Recogn Artif Intell (Spec Issue VLSI Algorithms Archit Comput Vis Image Process Pattern Recogn AI) 9(2):201–229
18. Chen E, Sabaz D, Gruver WA, Shannon L (2008) Replacement of the fixed multi-PR region model with a flexible, dynamic PR domain for DPR systems. In: Proc of international conference on field programmable technology
19. Mbock EAM (2013) Dynamic Partial Reconfiguration based on the Kalman filter method. In: Proc. of iasted on control and applications, Honolulu. http://www.actapress.com/Content_of_Proceeding.aspx?proceedingid=753. Accessed 02 Oct 2013
20. Mbock EAM (2012) Algorithms in matlab, the reality of abstraction and the power of pseudocodes, optimus verlag. http://www.mathworks.de/support/books/index_by_categorytitle.html?category=6. Accessed 31 Dec 2012
21. Higham NJ (1996) Accuracy and stability of numerical algorithms. Siam, Philadelphia
22. Skeel RD, Keiper JB (1993) Elementary numerical computing with mathematica. Mcgraw-hill, New York