Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.

This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/copyright

Author's personal copy

Available online at www.sciencedirect.com
ScienceDirect
Int. J. Electron. Commun. (AEÜ) 63 (2009) 398–405
International Journal AEÜ of Electronics and Communications
www.elsevier.de/aeue

Partial error tolerance for bit-plane FIR filter architecture

Vladimir Ćirić, Jelena Kolokotronis, Ivan Milentijević*

Faculty of Electronic Engineering, University of Nis, Aleksandra Medvedeva 14, P.O. Box 73, 18000 Nis, Serbia

Received 16 November 2007; accepted 22 February 2008

Abstract

Whereas some applications require correct computation many others do not. A large domain where perfect functional performance is not always required is multimedia and DSP systems. Relaxing the requirement of 100% correctness for devices and interconnections may dramatically reduce costs of manufacturing, verification, and testing. The goal of this paper is to develop a method for trading computational correctness for an additional chip area involved by fault-tolerance implementation. The method is demonstrated for the BP array in the following way: only the most significant bits of the output word are made fault-tolerant. By introducing the concept of partially error-tolerant BP array, designers achieve one more degree of tradeoff freedom. Formal definitions of the proposed terms are given. A mathematical path based on transitive closure that generates an error significance map for the BP array is proposed. The design tradeoff is demonstrated through FPGA implementation. The achieved area savings are presented as a function of a number of most significant fault-tolerant bits.
© 2008 Elsevier GmbH. All rights reserved.

Keywords: Error tolerance; Fault tolerance; Systolic arrays; FIR filtering

1. Introduction

As scaling approaches the physical limits of devices and fabrication technology, designers will increasingly have to consider qualitative changes. The key concerns include increasing process variations, defect rates, and infant mortality rates [1]. As VLSI scaling continues along its traditional path, we will soon be in a situation where chips will have billions of devices and thousands of defects [2,3].

Fault tolerance (FT) is the property that enables a system to continue operating properly in the event of failure of some of its components, at the cost of hardware, time or information quantity [4], which, in some cases, cannot be justified [5]. Relaxing the requirement of 100% correctness for devices and interconnections may dramatically reduce costs of manufacturing, verification, and testing. Such a paradigms shift is in any case forced by technology scaling, which leads to more transient and permanent failures of signals, logic values, devices, and interconnections [5].

In multimedia designers take advantage of the signal processing ability of people to convert the original source of signals to lower quality packets of information, since this usually provides acceptable performance to the end user, reduces bandwidth and hardware costs [6]. An interesting question is: if some signal processing device has a minor hardware defect, will it still produce results that are good enough for the end user? If so, they could also be sold rather than be discarded [7].

The finite impulse response (FIR) filtering is one of the important special purpose arithmetic operations widely used for video rate digital filtering. Variety of approaches for customizing implementation of FIR filters has been pursued. The bit-plane (BP) architecture is semi-systolic architecture with BP operations that provides regular connections with extensive pipelining and high computational throughput, which is, due to regularity, suitable for VLSI

* Corresponding author.
E-mail address: milentijevic@elfak.ni.ac.yu (I. Milentijević).

1434-8411/$ - see front matter © 2008 Elsevier GmbH. All rights reserved.
doi:10.1016/j.aeue.2008.02.015

Author's personal copy

implementations, either as a stand-alone module or as a part of complex digital data path [8,9].

In this paper we will investigate possibilities of trading off the acceptable computation correctness for hardware size in FIR filter design based on BP array. The goal of this paper is to develop a method for trading computational correctness for an additional chip area involved by FT implementation. This will be achieved by making only the most significant bits of an output word of the BP array fault-tolerant. By relaxing the requirement of 100% chip correctness, and enabling the tradeoff, only the marked cells will be implemented as FT cells.

We will start the development with acceptable margins of error in the resulting word, and from the transitive closure of the BP array we will obtain the error significance map. The array cells out of the area marked by the error significance map could produce errors, but without a significant influence on high order bits of the resulting word. Further on, formal definitions of proposed terms will be given. A rigorous mathematical path based on transitive closure that generates error significance map for the BP array will be proposed. The design tradeoff will be demonstrated through FPGA implementation. The achieved area savings will be presented as a function of a number of most significant fault-tolerant bits. By introducing the concept of partially error-tolerant (ET) BP array, one more degree of tradeoff freedom for designers will be involved.

The paper is organized as follows: Section 2 gives a brief overview of error tolerance; Section 3 is devoted to the architecture of BP FIR filter; in Section 4 we give definitions of basic terms; in Section 5 we propose the transitive closure of the BP array; Section 6 gives an example of an error significance map development. In Section 7 implementation results are presented, while in Section 8 concluding remarks are given.

2. Error tolerance

In order to introduce a tradeoff between an acceptable computation correctness and hardware size in FIR filter design based on a BP array, in this section we will give a brief overview of error and FT.

Error detection is the ability to detect the presence of errors, while error correction is the additional ability to reconstruct the original, error-free data.

Error-free operation may be performed by a block that can be reconfigured so that it outputs no errors. However, its result is the degradation of one or more of the block attributes. This is often called a graceful degradation [1]. Application of such a scheme to a system is usually referred to as a fault tolerant system. FT is the property that enables a system to continue operating properly in the event of failure of some of its components [4].

Spare components in a FT scheme refer to the first fundamental characteristic of FT—not single point of failure. Triple modular redundancy (TMR) is a fault tolerant form of N-modular redundancy in which three systems perform a process whose result is processed by a voting system to produce a single output. This means that if one of the three systems fails, the other two systems can correct and mask the fault. If the voter fails, then the complete system fails. However, in a good TMR system the voter is much more reliable than the other TMR components [4].

Error tolerance is an alternative concept. Error tolerant systems neither detect nor correct error. The circuit is said to be an ET, with respect to an application, if (1) it contains defects that cause internal and may cause external *errors*, and (2) the system that incorporates this circuit produces *acceptable results* [1,2,7].

There are passive and active measures of a system degradation due to errors. Performance, capacity and throughput are said to be passive measures, while error Rate, Accumulation and Significance, known as RAS, are active measures [1].

FT schemes, like TMR, are area consumptive. Thus, in order to design a system that trades an acceptable computation correctness for a chip area consumed by redundant cells, we propose combining TMR and ET.

The next section gives a brief review of the architecture of a BP FIR filter that is taken as a basis for implementation of BP FIR filter partially tolerant to errors.

3. Bit-plane FIR filter architecture

Output words $\{y_i\}$ of an FIR filter are computed as

$$y_i = c_0 x_i + c_1 x_{i-1} + \cdots + c_{k-1} x_{i-k+1}, \tag{1}$$

where $c_0, c_1, \ldots, c_{k-1}$ are coefficients while $\{x_i\}$ are input words. Computation (1) can be realized in different manners. When high performances are required systolic arrays are frequently used. Semi-systolic array shares with systolic arrays not only the desirable simplicity and regularity properties, but also pipelining and multiprocessing schemes of operation.

The BP is a semi-systolic architecture with BP operations. It provides regular connections with extensive pipelining and high computational throughput [8,9]. A functional block diagram of a BP array is shown in Fig. 1.

The following notation is adopted: $\boldsymbol{m}$—coefficient word length; $\boldsymbol{k_C}$—number of coefficients $(c_0, c_1, \ldots, c_{k_C-1})$; $\boldsymbol{n}$—input word length; $\boldsymbol{c_i^j}$—bit of coefficient c_i (with weight 2^j); $\boldsymbol{c_i} \equiv c_i^{m-1} c_i^{m-2} \cdots c_i^0$, where $c_i^0 c_i^1 \ldots c_i^{m-1}$ are the bits of coefficient c_i with weights $2^0, 2^1, \ldots, 2^{m-1}$, respectively; $\boldsymbol{c^j} \equiv c_{k-1}^j c_{k-2}^j \ldots c_0^j$, where $c_0^j, c_1^j, \ldots, c_{k-1}^j$, are the bits with weight 2^j of coefficients $c_0, c_1, \ldots, c_{k-1}$, respectively; $\boldsymbol{l_0}$—the number of basic cells within one row of a BP array; $\boldsymbol{y_i^j}$—the bit of output word y_i with weight 2^j.

There are m BP elements that form the array shown in Fig. 1. Each BP (Fig. 1) is formed as a set of k_C rows. A row

Author's personal copy

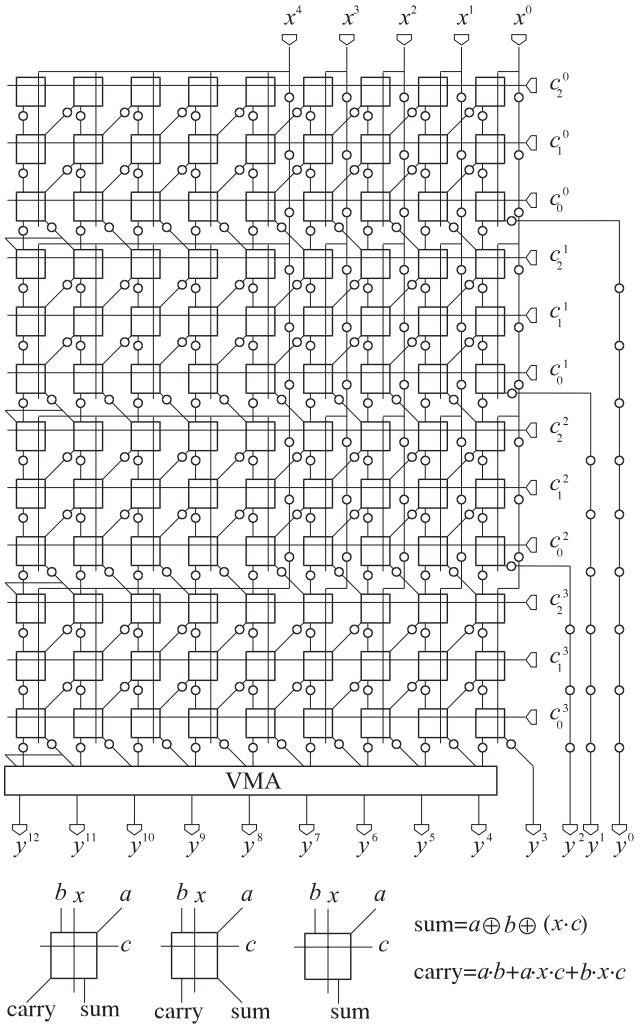

Fig. 1. The BP array for $k_C = 3$ and $m = 4$.

performs the basic multiply-accumulate operation between the intermediate result from the previous row and the product of the input word and one coefficient bit. Delayed for one clock cycle per row, the output word is available after $k_C \cdot m$ clock cycles.

4. Error significance and partial error-tolerance

In order to introduce a partial error-tolerance in a formal way, let us define basic terms.

Let the architecture be represented by the data flow graph **G**. Given a directed graph $\mathbf{G}=(\mathbf{V}, \mathbf{E})$, where $\mathbf{V}=\{v_1, \ldots, v_n\}$ is a finite set of vertices and **E** is a finite set of edges. An edge $e \in \mathbf{E}$ is an ordered pair (v_i, v_j), where $v_i, v_j \in \mathbf{V}$ and an edge (v_i, v_j) means that vertices v_i and v_j are connected.

Definition 1 (*Error propagation*). Let v_i and v_j be vertices and let $e_{i,j}$ denote $(v_i, v_j) \in \mathbf{E}$. We define an error propagation as the relation

$$\xi \subseteq \mathbf{V}^2, \quad (v_i, v_j) \in \xi.$$

The fact $(v_i, v_j) \in \xi$ we denote by $\xi_{i,j}$, which holds if an error which accrues within node v_i causes an error within an error-free node v_j.

Lemma 1. *Error propagation ξ is a transitive relation*:

$$v_i, v_k, v_j \in V, \xi_{i,k} \wedge \xi_{k,j} \Rightarrow \xi_{i,j}.$$

Proof. If $\xi_{i,k}$ and $\xi_{k,j}$ hold, the error within node v_k, caused by error in node v_i, will produce error in node v_j, which proves that nodes v_i and v_j are in relation ξ. □

Definition 2 (*Error significance*). Let $Y = \{y^0, y^1, \ldots, y^{l_0-1}\}$, $Y \subseteq \mathbf{V}$ be the set of architecture's output nodes. We call the set $\mathbf{M}_\eta \subseteq \mathbf{V}$ error significance for the output bit y^η, iff

$$v_i \in \mathbf{M}_\eta \Leftrightarrow (v_i, y^\eta) \in \xi.$$

Error significance (Definition 2) marks the part of the architecture that has to be error-free in order to have the output result bit y^η error-free.

Let two-dimensional ordering of an architecture be defined as a function $f_o: \mathbf{V} \to \mathbf{N}^2$, where **N** is a set of natural numbers. We call the function f_o the ordering function.

Definition 3 (*Error significance map*). We define error significance map, $M_\eta = (m^\eta_{p,q})$, for the output bit y^η, as a matrix with elements

$$m^\eta_{p,q} = \begin{cases} 1, & \exists\, v_i \in \mathbf{M}_\eta,\ f_o(v_i) = (p, q), \\ 0, & \forall\, v_i \in \mathbf{M}_\eta,\ f_o(v_i) \neq (p, q). \end{cases}$$

Definition 4 (*Partial error-tolerance*). We define a degree of partial error-tolerance (PET) of an architecture as a function from the set $\{0, 1, \ldots, l_0 - 1\}$ into the subset of **V**, such that

$$\mathbf{P}_{\mathrm{ET}}(\alpha) = \bigcup_{\eta=l_0-\alpha}^{l_0-1} \mathbf{M}_\eta.$$

Obviously, for $\alpha = 0$ union in Definition 4 is an empty set. Thus, $\mathbf{P}_{\mathrm{ET}}(0)$ is the basic architecture without FT. In respect of Definition 4, $\mathbf{P}_{\mathrm{ET}}(l_0)$ is a full fault tolerant (FFT) architecture.

Having in mind that error propagation is a transitive relation (Lemma 1), the error significance map of the BP array can be obtained from transitive closure, which gives information about all paths within the array.

5. Transitive closure of the bit-plane array

In order to clarify the error significance map development (Definition 3), we give a brief review of transitive closure [10,11].

Author's personal copy

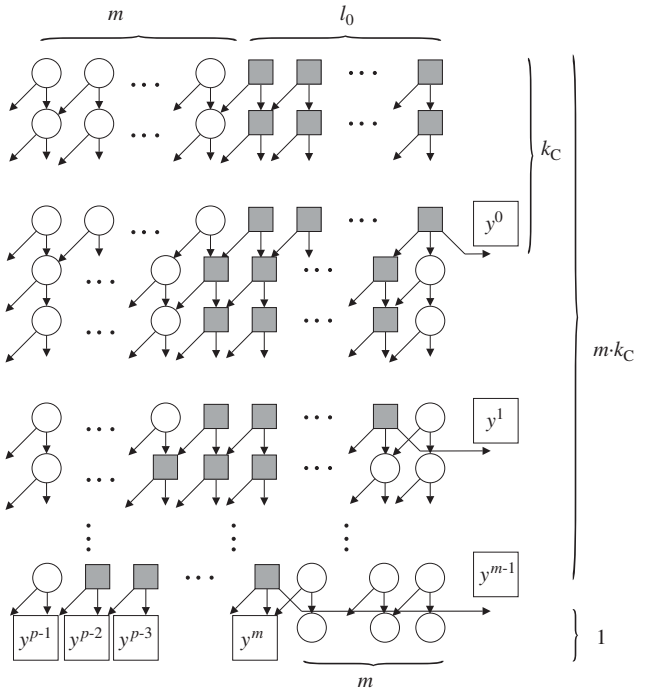

Fig. 2. Directed graph **G** for BP array.

The transitive closure of **G** is defined as the graph $\mathbf{G}^* = (\mathbf{V}, \mathbf{E}^*)$, where $\mathbf{E}^* = \{(v_i, v_j)|$ there is a path from v_i to v_j in **G**$\}$. The transitive closure of a graph is obtained by computing the connectivity matrix $\mathbf{A}^*$. The connectivity matrix of **G** is a matrix $\mathbf{A}^* = (a^*_{i,j})$ where $a^*_{i,j} = 1$ if there is a path from v_i to v_j or $i = j$, and $a^*_{i,j} = 0$ otherwise [10,11].

In respect of Definition 2, error significance $\mathbf{M}_\eta$ can be obtained from the column of transitive closure $\mathbf{A}^*$ that corresponds to the output node y^η. We define the error significance map M_η, according to Definition 3, for the BP architecture using the ordering function f_o, which maps the elements $a^*_{i,j}$ into $m^\eta_{p,q}$ as follows:

$$p = \left\lfloor \frac{j}{l_0 + m} \right\rfloor,$$

$$q = j \bmod (l_0 + m), \tag{2}$$

where $i = c^{ta}$ and $0 \leqslant j \leqslant (m \cdot k_C + 1) \cdot (m + l_0) - 1$.

In order to develop a transitive closure for the BP array, both the directed graph **G** for the array in Fig. 1, and the connectivity matrix $\mathbf{A}^*$ have to be obtained. In the BP array shown in Fig. 1 there is a multiplication of a partial product from each row within BP with factor 2, which is realized as a shift for one position to the left between rows. Between BPs there is a multiplication of a partial product by $\frac{1}{2}$, i.e., shift for one position to the right (Fig. 1), which introduces slight irregularities between rows [8]. In order to develop connectivity matrix for BP array in a general form, according to the functional block diagram in Fig. 1, we obtained a directed graph **G** shown in Fig. 2.

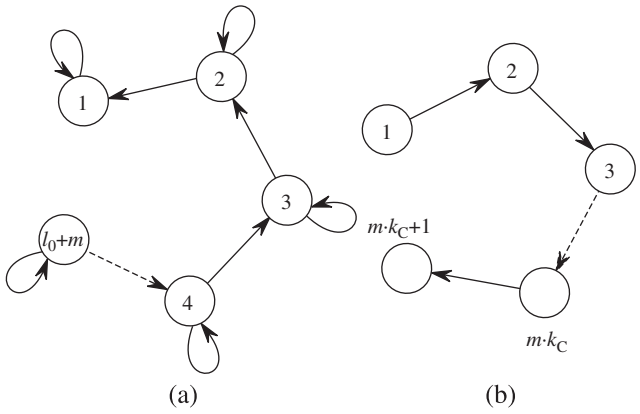

Fig. 3. Directed graph: (a) $\mathbf{G_C}$ that shows the connectivity of columns of graph **G**, and (b) graph $\mathbf{G_R}$ that shows connectivity of rows.

Graph **G** in Fig. 2 consists of two types of nodes: nodes that represent the BP cells in Fig. 1 (shaded nodes in Fig. 2), and fictive nodes that are added in order to obtain a graph with regular connections. In order to obtain connectivity matrix of graph **G** we form a directed graph $\mathbf{G_C}$ that shows the connectivity of columns of graph **G**, and graph $\mathbf{G_R}$ that shows the connectivity of rows. Graphs $\mathbf{G_C}$ and $\mathbf{G_R}$ are shown in Figs. 3(a), and (b), respectively.

Graph in Fig. 3(a) can be represented by connectivity matrix as follows:

$$G_C = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 0 & 1 & 1 & \dots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}_{(l_0+m)\times(l_0+m)}.$$

The dimension of matrix G_C is $(l_0 + m) \times (l_0 + m)$, where $l_0 + m$ is the number of columns of graph **G** (Fig. 2). The elements $g^C_{i,j}$ of matrix G_C are of the form:

$$g^C_{i,j} = \begin{cases} 1, & j + 1 \geqslant i \geqslant j, \\ 0, & \text{other} \end{cases} \tag{3}$$

that is, the elements on the main diagonal, and the elements below the main diagonal are equal to 1. Transitive closure $\mathbf{A}^*$ can be obtained from graph $\mathbf{G_R}$ (Fig. 3b), because each node in graph $\mathbf{G_R}$ has internal paths described with (3). The connectivity matrix G_R for the directed graph **G** in Fig. 2 is

$$G_R = \begin{bmatrix} 0 & G_C & 0 & \dots & 0 \\ 0 & 0 & G_C & \dots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & G_C \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}_{(m\cdot k_C+1)\times(m\cdot k_C+1)}. \tag{4}$$

Substituting G_C in (4) with (3), we derive a connectivity matrix for graph **G** from Fig. 2. The transitive closure $\mathbf{A}^*$,

Author's personal copy

obtained using (4), is of the following form:

$$\mathbf{A}^* = \begin{bmatrix} 0 & G_C & G_C^2 & \dots & G_C^{m\cdot k_C} \\ 0 & 0 & G_C & \dots & G_C^{m\cdot k_C-1} \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & G_C \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \tag{5}$$

where $G_C^k = G_C^{k-1} \cdot G_C$. In order to obtain matrices G_C^k from transitive closure (5), we give the following lemma:

Lemma 2. *Elements* $((g_{i,j}^C)^d)$ *of the matrix* G_C^d *are of the form*

$$(g_{i,j}^C)^d = \begin{cases} 1, & j + d \geqslant i \geqslant j \\ 0, & \text{other} \end{cases}.$$

Proof. Lemma 2 can be proven using mathematical induction. Elements $(g_{i,j}^C)^1$ of the matrix G_C^1 are of the form (3). From (3) it follows directly that Lemma 2 stands for G_C^2. In order to show that Lemma 2 stands for G_C^{d+1}, we obtain $(g_{i,j}^C)^{d+1}$ as follows:

$$\begin{aligned} G_C^{d+1} = G_C^d \cdot G_C \Rightarrow (g_{i,j}^C)^{d+1} &= \sum_{k=0}^{p} (g_{i,k}^C)^d \cdot g_{k,j}^C \\ &= \begin{cases} 1, & k + d \geqslant i \geqslant k \bigwedge j + 1 \geqslant k \geqslant j \\ 0, & \text{other} \end{cases} \\ &= \begin{cases} 1, & j + d + 1 \geqslant i \geqslant j \\ 0, & \text{other} \end{cases}, \end{aligned}$$

which proves Lemma 2. □

Transitive closure $\mathbf{A}^*$, given with (5), is a square matrix with the dimension $(m\cdot k_C+1)\cdot(m+l_0)\times(m\cdot k_C+1)\cdot(m+l_0)$, which shows the existence of the path between any two nodes of the graph **G** (Fig. 2). Each column of the transitive closure (5) that corresponds to an output node stands for one error significance map M_η of the BP array (Definition 3, Eq. (2)).

6. Example of partial error-tolerant bit-plane array

With the aim to illustrate partial ET BP array according to Definition 4, using the transitive closure of BP array (5) and an error significance map M_η (Definition 3) defined with ordering function f_o (2), we give an example of BP array with $k_C = 2$ coefficients, and coefficient length $m = 2$.

A directed graph **G** of BP array with $k_C = 2$ coefficients, and coefficient length $m = 2$ is shown in Fig. 4(a). A transitive closure of the graph is obtained according to (5) and Lemma 2, and is shown in Fig. 4(b). The dimensions of $\mathbf{A}^*$ for the given example are 30 × 30 (enumerated as 0, 1, . . . , 29), because there is a total of 30 nodes within the graph **G**, including the output nodes y^5, y^4, y^3, and y^2. Fig. 4(b) shows the upper-right corner of transitive closure $\mathbf{A}^*$ of the graph **G** in Fig. 4(a).

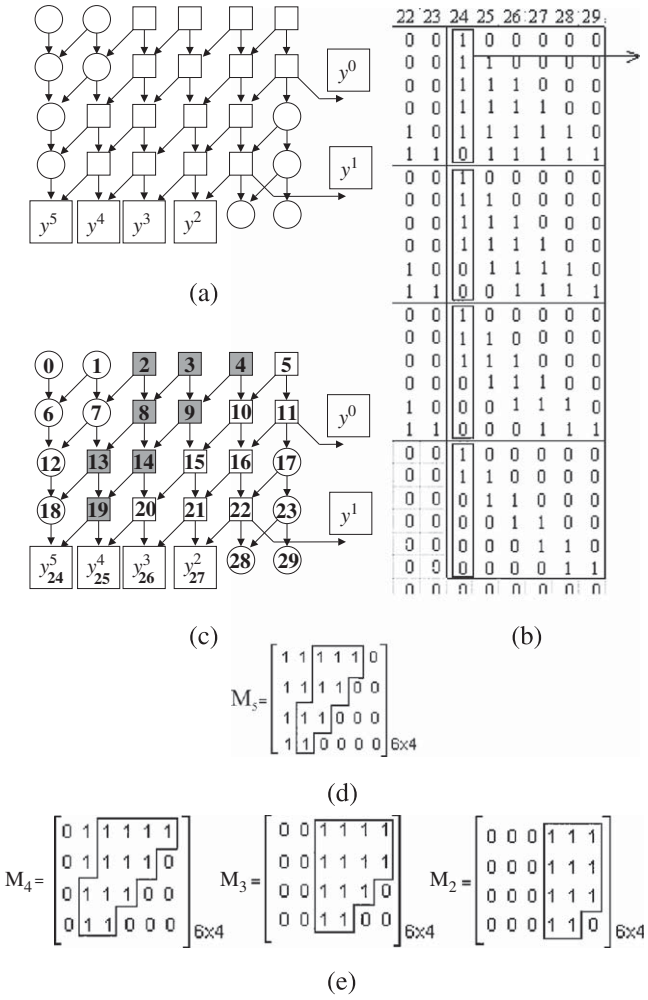

Fig. 4. Implementation example: (a) directed graph **G** of BP array with $k_C = 2$ coefficients, and coefficient length $m = 2$; (b) transitive closure $\mathbf{A}^*$; (c) error significance set $\mathbf{M}_5$; (d) error significance map M_5; (e) error significance maps M_4, M_3, and M_2.

A transitive closure, given in Fig. 4(b), shows the existence of all paths within the graph. According to Definition 4, $\mathbf{P}_{\mathrm{ET}}(1)$ can be obtained applying the FT scheme to the cells marked by error significance map M_5. The error significance map M_5 can be obtained, according to (2), from transitive closure $\mathbf{A}^*$ by rewriting the 24th column, which corresponds to the output bit y^5, Fig. 4(c). The nodes of the graph from Fig. 4(a) are enumerated as 0, 1, . . . , 29, as shown in Fig. 4(c). The elements of the 24th column of $\mathbf{A}^*$ are rewritten in the form of a matrix, according to (2), and mapped to the nodes of graph **G**, which is shown in Fig. 4(c). The nodes that have influence on the most significant bit of the result are shaded. The fictive nodes, because of their nature, are not taken into consideration. Fig. 4(d) shows the error significance map M_5 of the BP array.

Author's personal copy

If acceptable results of BP array with $k_C = 2$, and $m = 2$ are defined as $Y_{\text{acceptable}} = Y_{\text{correct}} \pm (2^5 - 1)$, then matrix M_5 in Fig. 4(d) shows the part of the BP array which must be error-free in order for the BP architecture to produce *acceptable results*.

Error significance maps M_4, M_3, and M_2 are developed in the same manner, and are shown in Fig. 4(e). For example, according to the Definition 4, $\mathbf{P}_{\text{ET}}(2)$ can be obtained making the cells, marked by M_5 and M_4, fault-tolerant.

7. Implementation results

In order to illustrate the tradeoffs enabled by partial employment of FT in the array, we choose a TMR as a FT scheme to be involved in PET BP array implementation (Definition 4).

For the sake of comparison, both BP architecture shown in Fig. 1, and $\mathbf{P}_{\text{ET}}(\alpha)$, $1 \leqslant \alpha \leqslant l_0$, were described in VHDL and implemented on Virtex4 FPGA. Design automation is involved using the concept of *generic* parameters in VHDL [12]. Function $\mathbf{P}_{\text{ET}}(\alpha)$ (Definition 4) is implemented as a custom-written VHDL function with α as a parameter which controls whether the cell should be triplicated or not in the moment of cell instantiation. Using the design automation described in VHDL, the only parameter which has to be set, regardless the array dimensions, is the number of error-free high order bits in the output word (α).

We implemented three arrays with different parameter sets. Parameter sets for the implemented arrays are given in Table 1. Values for arrays widths (l_0), and heights ($m \cdot k_C$) are given in bold. The arrays differ in input word width (n), which implies the different array widths and the same array heights. All three arrays are implemented for parameter α in the range $0 \leqslant \alpha \leqslant l_0$.

Implementation results are given in Table 2. For each array from Table 1 there are two columns within Table 2. The left column stands for the number of basic cells obtained analytically using the proposed method, while the right column stands for FPGA implementation in equivalent number of kilogates (kG) [12].

Let us explain the analytically obtained results for *Arr1*. If there are no ET bits in the output result ($\alpha = 0$), then the array is basic BP array (Fig. 1), and the number of the required basic cells is equal to $(m \cdot k_C) \cdot l_0 = 8 \cdot 4 \cdot 16 = 512$. If all bits of the output result are ET ($\alpha = l_0 = 16$), FFT array, then all the basic cells have to be triplicated

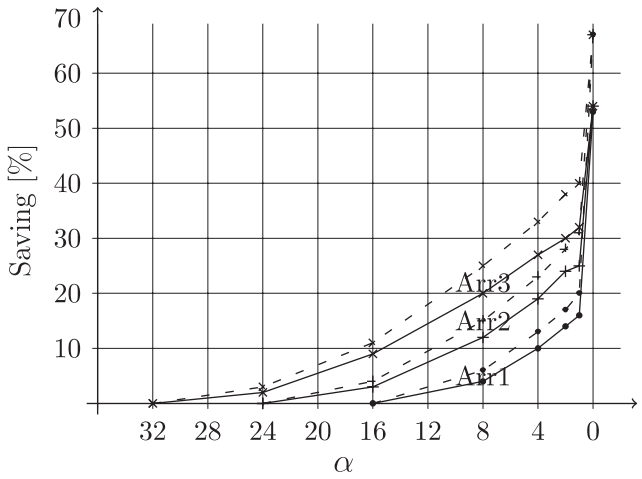

Fig. 5. Saving of the silicon area required for array implementation as the function of bits in the output word that are error-tolerant (α).

Table 1. Parameter sets for implemented arrays

	k_C	m	n	l_0	$m \cdot k_C$	$\mathbf{P}_{\text{ET}}(\alpha)$
Arr1	4	8	8	**16**	**32**	$0 \leqslant \alpha \leqslant 16$
Arr2	4	8	16	**24**	**32**	$0 \leqslant \alpha \leqslant 24$
Arr3	4	8	24	**32**	**32**	$0 \leqslant \alpha \leqslant 32$

Table 2. Number of basic cells required for array implementation, and actual gate count by FPGA implementation of different sizes of arrays with α as parameter

α	Arr1		Arr2		Arr3	
	No. of cells	Impl. [kG]	No. of cells	Impl. [kG]	No. of cells	Impl. [kG]
32	/	/	/	/	**3072**	**82.0**
24	/	/	**2304**	**61.6**	2986	80.2
16	**1536**	**41.4**	2218	59.8	2730	74.7
8	1450	39.6	1962	54.3	2304	65.4
4	1344	37.3	1770	50.1	2048	59.9
2	1274	35.8	1658	47.1	1920	57.1
1	1236	34.9	1598	46.4	1856	55.7
0	512	19.4	768	28.6	1024	37.8

Author's personal copy

($512 \cdot 3 = 1536$, Table 2). For $\alpha = 1$, according to Definition 4, 362 (out of 512) basic cells have to be triplicated, thus $362 \cdot 3 + (512 - 362) \cdot 1 = 1236$ basic cells are required for the array implementation (Table 2).

The number of basic cells, as well as the number of gates required for the implementation of FFT arrays, are given in bold in Table 2. These values are used for the calculation of achieved savings in the implementation of ET arrays with $\alpha < l_0$, which are graphically shown in Fig. 5. For both the analytically obtained and FPGA implementation results savings are calculated as

$$\text{Saving} = \frac{\text{FFT} - \mathbf{P}_{\text{ET}}(\alpha)}{\text{FFT}} \cdot 100\%. \quad (6)$$

The analytically obtained savings are shown with dashed lines, while FPGA implementation results are shown with solid lines in Fig. 2. It can be noticed that FPGA implementation results are slightly shifted below the corresponding theoretical results. This is due to the optimization effort of a VHDL synthesis tool. It can also be noticed that for $\alpha = 0$ all the theoretical results end with the saving equal to 66.7 (Fig. 5). According to (6), and having in mind that for TMR $\mathbf{P}_{\text{ET}}(0) = \text{FFT}/3$, "the saving", if none of the result bits is ET, is $(\text{FFT} - \text{FFT}/3)/\text{FFT} \cdot 100\% = 66.7\%$. FPGA implementation results end with saving of 56%.

8. Concluding remarks

In this paper we proposed a concept of systems partially tolerant to errors, and presented the design of a partially ET BP array. Partial tolerance to errors is employed allowing some of the cells to produce errors. The design of partially ET BP FIR filter is facilitated by deriving the error significance map for the BP array. Starting the development with acceptable margins of error in the output result, we obtained the error significance map from the transitive closure of the BP array. The array cells out of the area marked by error significance map could produce errors, but without any significant influence on the high-order bits of the resulting word. Formal definitions of the proposed terms are given. A rigorous mathematical path based on transitive closure that generates error significance map for the BP array is proposed. The design tradeoff is demonstrated through an FPGA implementation. The achieved area savings are presented as a function of a number of most significant fault-tolerant bits. By introducing the concept of partially ET BP array, designers achieve one more degree of tradeoff freedom.

References

[1] Breuer M, Gupta S, Mark T. Defect and error tolerance in the presence of massive numbers of defects. IEEE Trans Design Test Comput 2004;21:216–27.

[2] Breuer M. Intelligible test techniques to support error-tolerance. In: Proceedings on the 13th asian test symposium (ATS 2004), IEEE Computer Society, 2004. 0–7695–2235–1/04.

[3] Hsieh T-Y, Lee K-J, Breuer M. Reduction of detected acceptable faults for yield improvement via error-tolerance. In: Proceedings of the conference on design, automation and test in Europe, Nice, France, 2007. p. 1599–1604.

[4] Johnson BW. Fault tolerance: the electrical engineering handbook. CRC Press; 1993.

[5] International Technology Roadmap for Semiconductors: Recommendations. ⟨http://public.itrs.net/⟩, 2001.

[6] Momcilovic S, Roma N, Sousa L. Adaptive motion estimation algorithm for h.264/avc. IEEE 15th international conference on digital signal processing (DSP) 2007, Cardiff, Wales, UK, July 2007.

[7] Breuer M. Multimedia applications and imprecise computation. In: Proceedings on the eighth Euromicro conference on digital system design, Euromicro, Porto, Portugal, September 2005.

[8] Noll T. Semi-systolic maximum rate transversal filters with programmable coefficients. Workshop of systolic architectures, Oxford, UK, 1986. p. 103–12.

[9] Ciric V, Milentijevic I. Configurable folded array for FIR filtering. J Syst Architect 2007, in press, doi:10.1016/j.sysarc.2007.05.001.

[10] Gries D, Schneider F. A logical approach to discrete math. Berlin: Springer; 1993.

[11] Bawa S, Sharma G. A parallel transitive closure computation algorithm for VLSI test generation. In: Proceedings PARA 2002, Espoo, Lecture Notes in Computer Science, vol. 2367, 2002. p. 243–52.

[12] Pedroni VA. Circuit design with VHDL. Massachusetts Institute of Technology, Massachusetts: MIT Press; 2004.

Vladimir M. Ciric is a teaching and research assistant at the Faculty of Electronic Engineering at the University of Nis, Serbia. He received both B.S. and M.Sc. degrees from the Faculty of Electronic Engineering, University of Nis, 2001 and 2005, respectively, where he currently attends the Ph.D. studies. His research interests include computer architectures, fast arithmetic, fault-tolerant systems, digital image processing and video coding.

Jelena Kolokotronis is a Ph.D. student at the Faculty of Electronic Engineering at the University of Nis, Serbia. She received graduated engineer diploma from the Faculty of Electronic Engineering, University of Nis in 2007. Her research interests include computer architectures and fault-tolerant systems.

Author's personal copy

Ivan Z. Milentijevic is an Associate Professor at the Faculty of Electronic Engineering, University of Nis, Serbia. He received B.S. in Electrical engineering and M.Sc. and Ph.D. degree in Computer Science from the Faculty of Electronic Engineering in 1989, 1994 and 1998, respectively. His research interests include computer architecture, parallel processing, fast and fault tolerant arithmetic, digital signal processing and computer science education. He published 20 journal papers and coordinated and managed 2 international projects. Currently he is head of the Computer Science Department at the University of Nis.