A Parallel Architecture for the ICA Algorithm: DSP Plane of a 3-D Heterogeneous Sensor

V. K. Jain[1] **S. Bhanja**[1] **G. H. Chapman**[2] **L. Doddannagari**[1] **N. Nguyen**[1]

[1] University of South Florida
Tampa, Florida 33620, U.S.A.

[2] Simon Fraser University
Burnaby, BC, Canada V5A 1S6

Abstract: A 3-D Heterogeneous Sensor using a stacked chip has recently been proposed. While the sensors are located on one of the planes, the other planes provide for analog processing, digital signal processing, and wireless communication. This paper[1] focuses on its DSP plane, in particular on the implementation of the ICA (Independent Component Analysis) algorithm in the DSP plane. ICA is a recently proposed method for solving the blind source separation problem. The objective is to recover the unobserved source signals from the observed mixtures without the knowledge of the mixing coefficients. We present a parallel architecture for it utilizing the reconfigurable J-platform, which employs *coarse-gain* VLSI cells with high functionality, performance, and reconfigurability. These include a Universal Nonlinear (UNL) cell, an extended multiply accumulate (MA_PLUS) cell, and a Data-Fabric (DF) cell. The coarse-grain approach has the distinct advantages of reduced external interconnect, much reduced design time, and manageable testability. Additionally, the other algorithms needed for the 3-D HSoC can also be mapped on to the same resources, by time multiplexing, thereby reducing the silicon area needed.

I. INTRODUCTION

A 3D Heterogeneous Sensor using a stacked chip has recently been proposed [1],[2]. While the sensors are located on the top plane, the other planes provide for analog processing, digital signal processing, and wireless communication. On the sensor plane four types of sensors are placed, namely visible imager (Active Pixel Sensor), infrared imager, seismic, and acoustic. The creation of integrated systems containing both sensors and processing power is of considerable interest in areas ranging from remote monitoring to security and defense. Ideally one would like an ultra-small, ultra-compact, unattended multi-phenomenological sensor system providing an integrated classification-and-decision-information extraction capability from the sensed environment. As illustrated in Fig. 1, the concept is to fabricate separate wafers for each plane (sensors, analog, digital processing, etc), mechanically polish the wafers to a thin structure, add inter-plane vias, separate into chip blocks and create a stacked 3-D structure. The target is to achieve a minimum 10X reduction in weight, volume, and power and a 10X or greater increase in capability and reliability – over alternative planar approaches. These gains will accrue from (a) the avoidance of long on-chip interconnects and chip-to-chip bonding wires, and (b) the cohabitation of sensors, preprocessing analog circuitry, digital logic and signal processing, and RF devices in the same compact volume. This concept is shown in Fig. 2 in greater detail, wherein a set of four types of sensors, namely an array of acoustic and seismic sensors, an active pixel sensor array, and an uncooled IR imaging array are placed on a common sensor plane. It is useful to remark that the *sensor set incorporates redundancy for defect tolerance*, and the DSP algorithm takes into account the corresponding changes in the locations of the sensors. More details can be found in [4].

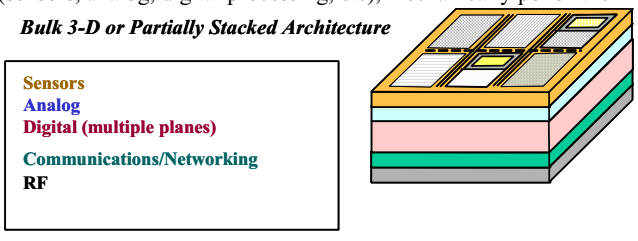

Fig. 1 3-D heterogeneous system on a chip

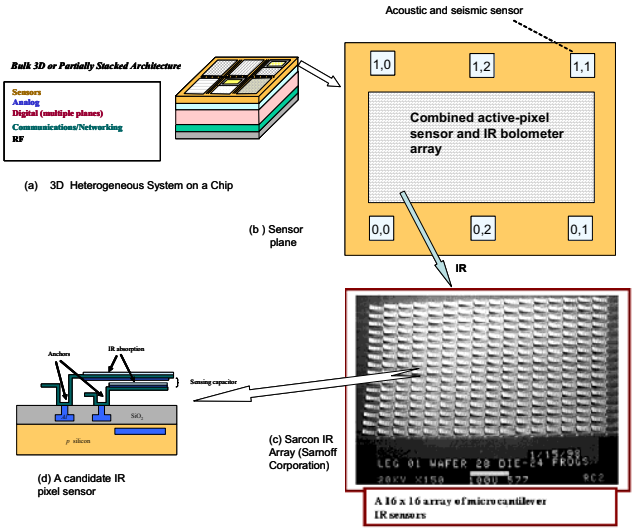

Fig. 2 Sensor plane for 3-D heterogeneous system

Among the planes mentioned above, the DSP plane provides for sensor data fusion, feature extraction and event detection/classification capability. Toward these goals, this paper focuses on the realization of the ICA (Independent Component Analysis) algorithm [3]-[4] on the DSP plane. ICA can be used for solving the blind source separation problem as well as for sensor data fusion. In the first case the objective is to recover the unobserved source signals from the observed mixtures without the knowledge of the mixing coefficients and in the second the extraction of the feature signals. It has the potential for a wide range of applications in industrial, medical, and security areas because it reduces the complex problem of dealing with high-dimensional

[1] This work was supported in part by NSF Grant No. 0441212.

statistical descriptions to products of one-dimensional density functions. In this paper we present a parallel architecture utilizing the reconfigurable J-platform [7], which employs *coarse-gain* VLSI cells with high functionality, performance, and reconfigurability. These include a Universal Nonlinear (UNL) cell, an extended multiply accumulate (MA_PLUS) cell, and a Data-Fabric (DF) cell [5]-[8]. The coarse-grain approach has the distinct advantages of reduced external interconnect, much reduced design time, and manageable testability. Additionally, the other algorithms needed for the 3D HSoC can also be mapped onto the same resources, by time multiplexing.

The paper is organized as follows. Section II discusses the motivation for employing the ICA algorithm. Section III very briefly describes the J-platform. Section IV focuses on the parallel architecture for the computations in the fast ICA algorithm, followed by estimates of performance in Section V. Time-multiplexing of resources is discussed in Section VI, and defect tolerance in Section VII.

II. The ICA and the Fast ICA

Imagine that in a room two people are speaking simultaneously and that there are two microphones which produce time signals denoted by $x_1(t)$ and $x_2(t)$. Each of these received signals is a weighted sum of the speech signals produced by the two speakers denoted by $s_1(t)$ and $s_2(t)$. Then we can express the received signals in terms of the original signals in terms of some weighting coefficients a_{11}, a_{12}, a_{21}, and a_{22} that depend on the microphone characteristics and their distances from the speakers. Clearly, it would be very useful to recover the original speech signals from the received signals. More generally, if there are m source signals and m received mixed signals, then their relationship can be expressed as

$$\begin{aligned} x_1(t) &= a_{11}\, s_1(t) + a_{12}\, s_2(t) + \ldots + a_{1m}\, s_m(t) \\ x_2(t) &= a_{21}\, s_1(t) + a_{22}\, s_2(t) + \ldots + a_{2m}\, s_m(t) \\ &\ldots \\ x_m(t) &= a_{m1}\, s_1(t) + a_{m2}\, s_2(t) + \ldots + a_{mm}\, s_m(t) \end{aligned} \quad (1)$$

or in matrix-vector notation $\underline{x}(t) = \boldsymbol{A}\, \underline{s}(t)$. Here, for example, s_1 and s_2 could be speech signals, s_3 could be the sound produced by a motor vehicle, etc. In a biomedical environment they could represent a set of EEG signals, ECG signals, blood pressure signals, etc.

The recently developed technique called ICA, can be used to estimate $\boldsymbol{A}$ or its inverse $\boldsymbol{W} = \boldsymbol{A}^{-1}$ based on the information of their statistical independence, which then allows blind separation of the original signals from their mixtures. The technique is applicable not only to time signals but also to images. As an example consider the three images s_1, s_2 and s_3 shown in the Fig. 3 (a), (b) and (c). Their histograms are shown in (d), (e) and (f). Suppose now that the observed images are the weighted mixtures shown in Fig. (g), (h) and (i). We now pose the question whether and how we can recover the original images blindly (without the knowledge of the mixing information). The answer is a 'yes' based upon certain mild assumptions which can be found in [1],[2]. Indeed, the images estimated by the application of the fast version of ICA, called Fast ICA [1], [2] are shown in Fig. 3 (j) (k) and (l). They are seen to be excellent replicas of the original images.

This algorithm has a wide range of potential applications in industrial and medical fields. For some applications, ICA analysis on a workstation may be adequate, but for many others it is desirable to have a VLSI chip that can perform independent component analysis (ICA) in real-time. In Section V we propose a parallel architecture for the real-time implementation of ICA algorithm on the reconfigurable J-platform, developed in our laboratory.

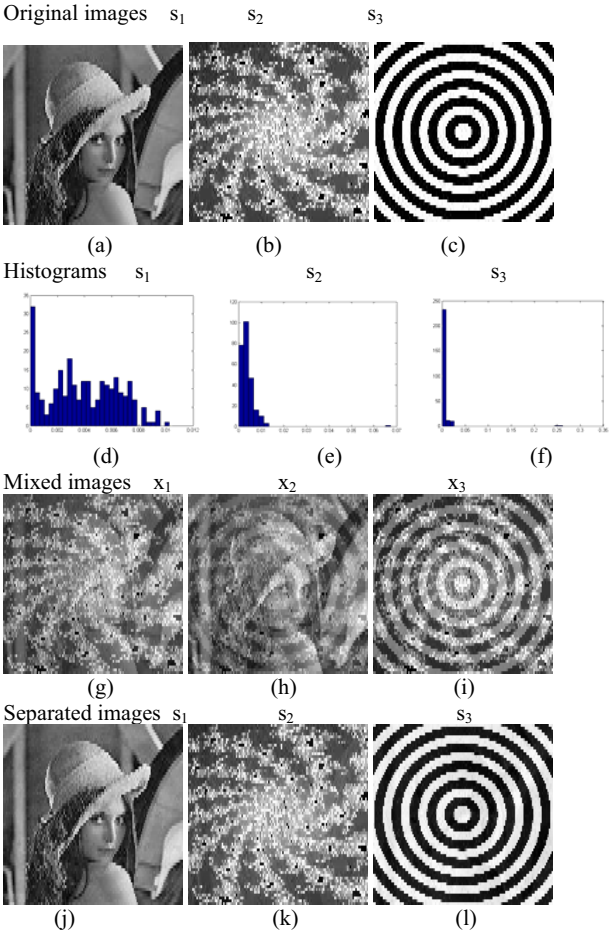

Fig. 3 Blind separation of the original images using ICA Algorithm

III. J-Platform

In recent years rapid system prototyping has attracted considerable attention. Newly emerging names for this technology include 'soft-hardware', 'structural software', and 'reconfigurable-computing'. The granularity at which the reconfiguration is performed can range from fine to medium, and medium to coarse. Based on coarse-grain cells, the J-Platform provides for many of the high speed applications such as FIR filtering of signal and images, Fast Fourier transform, Solution of a linear system of equations, and advanced applications like Reconstruction of CT images from fan beam projections and RGB to HSI conversion for video. The three very flexible cells on this platform can be used to map ultra high speed objectives with high performance. These cells are the MA_PLUS cell [7],[8] the Universal NonLinear (UNL) cell [5],[6], and the Data fabric cell [7],[8]. The MA_PLUS cell is a generalized multiply accumulate cell which can perform any of several operations in a highly efficient manner. An example is the accumulation of the

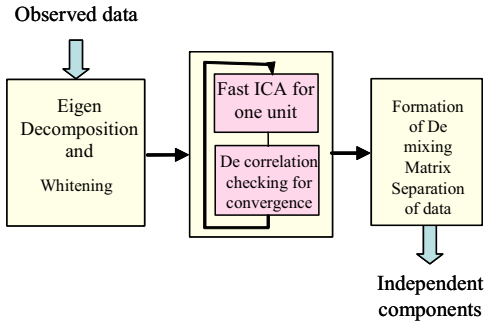

Fig. 4 Behavioral-level block diagram for ICA Algorithm

absolute difference in a single cycle, a computation which is pervasive in video encoders. The UNL cell can generate nonlinear functions, on a selectable basis *in a single cycle*. The data fabric cell can perform data routing, register buffering, and some minor computations.

IV. Fast ICA Pipelined Systolic Architecture

Before applying the ICA algorithm on the given data, it is useful to perform preprocessing. Some preprocessing techniques that can make the problem of ICA estimation simpler and better conditioned are centering, whitening and band pass filtering.

A. Whitening: Whitening reduces the number of parameters to be estimated. Whitened data $\tilde{x}$ has its components uncorrelated and their variances equal to unity. In other words, the covariance matrix of $\tilde{x}$ is an identity matrix. Whitening can be done using eigenvalue decomposition (EVD) of the covariance matrix of mixed signals $\underline{X}$, $\boldsymbol{C}$. Let $\boldsymbol{V}$ be the matrix of eigenvectors of $\boldsymbol{C}$ and $\boldsymbol{\Lambda}$ the diagonal matrix of eigenvalues of $\boldsymbol{C}$.

Whitening is done by

$$\tilde{x} = \boldsymbol{\Lambda}^{-1/2} \boldsymbol{V}^{T} x \qquad (2)$$

Whitening transforms mixing matrix A into $\tilde{A}$ where $\tilde{A} = \Lambda^{-1/2} V^T A$.

A parallel architecture for whitening is shown in Fig. 5. For simplicity of notation, hereafter we will drop the tilde on $\underline{x}$.

B. Iterative Computation: The fast ICA finds a direction, i.e. a unit vector $\underline{w}$ such that the projection $\underline{w}^T\underline{x}$ maximizes non-gaussianity. Non-gaussianity is measured by the negentropy $J(\underline{w}^T\underline{x})$ [1],[2] where

$$J(u) \approx [\, E\{G(u)\} - E\{\, G(z)\}\,]^2 \qquad (3)$$

Here, z is a zero mean and unit variance Gaussian variable. The variance of $\underline{w}^T\underline{x}$ has to be unity for the measure to be valid. For whitened data, this is equivalent to constraining the norm of w to be unity. To prevent different vectors from converging to the same maxima, we must decorrelate them after each iteration. For this, when we have estimated p vectors, $\underline{w}_1$, $\underline{w}_2$, …, $\underline{w}_p$, we run the algorithm for $\underline{w}_{p+1}$, and after every iteration step subtract from $\underline{w}_{p+1}$ the projection matrix BB^T formed from the matrix B whose columns are $\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_p$.

The algorithm consists of the following steps:

Step 1: *Initialization:* Choose initial random weight vector $\underline{w}_n(0)$ with norm 1. Let B be the null matrix of the size of number of independent components.

Step2: *Iteration:* Let the non-linear function be

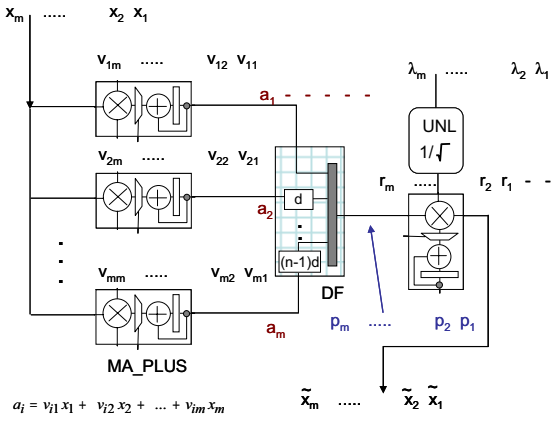

Fig. 5 Parallel architecture for whitening

$$G(u) = -\exp(-u^2/2) \qquad (4)$$

Then

$$g(u) \triangleq G'(u) = u \exp(-u^2/2) \qquad (5)$$

$$g'(u) = (1-u^2)\exp(-u^2/2)$$

The update of the n-th row of $\boldsymbol{W}$ is given by $\underline{w}_n(k+1)^T$.

$$\underline{w}_n(k+1) = E\{\underline{x}\, g(\underline{w}_n(k)^T \underline{x}) - E\{g'(\underline{w}_n(k)^T \underline{x})\}\, \underline{w}_n(k)\} \qquad (6a)$$

$$\underline{w}_n(k+1) <= \frac{\underline{w}_n(k+1)}{\| \underline{w}_n(k+1) \|} \qquad (6b)$$

The key steps of the fast ICA algorithm, namely the step in (4) through (6), can be mapped onto the J-platform as shown in Fig. 6. We will call this block as the *main block*. Note that a total of K sensed data vectors, each of dimension m are fed from the top. The normalized updated weight vector appears at the output (bottom right).

***Step 3*: *Decorrelation*:**

To prevent the different vectors from converging to the same maxima, it needs to be decorrelated.

$$\underline{w}_n(k+1) = \underline{w}_n(k+1) - BB^T \underline{w}_n(k+1) \qquad (7)$$

$$\underline{w}_n(k+1) = \frac{\underline{w}_n(k+1)}{\| \underline{w}_n(k+1) \|} \qquad (8)$$

Step 4: If $\underline{w}_n(k+1)$ and $\underline{w}_n(k)$ have converged, then go to step 5, else increment k to $k+1$ and go to step 2. $\underline{w}_n(k+1)$

***Step 5*:** Replace the n-th column of $\boldsymbol{B}$ with $\underline{w}_n(k+1)$. After whitening, $\underline{w}_n(k+1)^T$ is added as the n-th row of $\boldsymbol{W}$. Increment n to $n+1$ and set $k = 0$. If n ≤ number of independent components, then go to step 2 else stop.

V. Estimated Performance

Consider that the various segments of the algorithm described above are pipelined. Then the performance is dominated by the segment, or block, that requires the maximum amount of computation time. That particular dominant block is the main block of Fig. 6. For a total of I iterations, and a total of K sensed data vectors, each of size m, the total number of cycles can be shown to be $I(10+m(k+1))$. Assuming the word length to be 16 bits and that the MA_PLUS and UNL operate at 600 MHz, the total time required for m=4, K=100, and I=10 can be shown to be 70 μs. If a factor of 10 time-multiplexing is

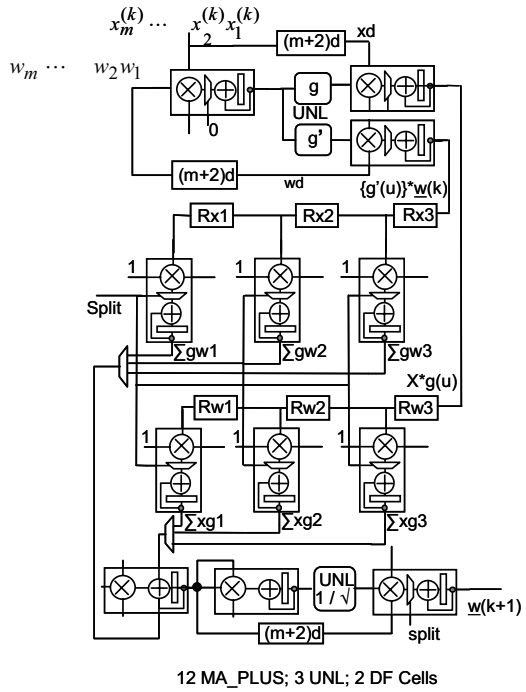

Fig. 6 Main block of the fast ICA algorithm (without time-multiplexing)

used to reduce the VLSI resources, then the time required would be ~700 μs.

VI. Time-Multiplexing on the J-Platform

Although not presented here due to a lack of space, the architectures for eigenvalue decomposition and decorrelation, using the powerful cells of the J-platform, are also quite interesting. Also important to note is the fact that all three types of cells, the MA_PLUS, the UNL, and the DF are very high speed cells, and can therefore be multiplexed in time for most sensor applications in order to reduce the hardware cost. Estimated speed for 16 to 32 bit cells ranges from 600 MHz to 500 MHz in 0.18 micron technology [10]. Thus, for a desired application speed of 20 MHz, each of these cells could impersonate 32 to 16 cells, but of course at the cost of some multiplexers and registers.

VII. Defect Tolerance Using Spares

Redundancy is provided by incorporating spare MA_PLUS, UNL, and DF cells. A brief analysis of the yield is given below.

Notation: $p_1, p_2, p_3,$ Pr{that an MA_PLUS cell, a UNL cell, or a DF cell is tested good}. For subalgorithm-i (such as the whitening), let $N_{i,1}$ = number of MA_PLUS cells provided; $N_{i,2}$ = number of UNL cells provided; $N_{i,3}$ = number of DF cells provided; $M_{i,1}$ = number of MA_PLUS cells required; $M_{i,2}$ = number of UNL cells required; $M_{i,3}$ = number of DF cells required. Then, the yield for that block is given by $Y_i = Y_{i,1}\, Y_{i,2}\, Y_{i,3}\, Y_{i,4}$, where

$$Y_{i,j} = \binom{N_{i,j}}{M_{i,j}} p_j^{M_{i,j}} (1-p_j)^{(N_{i,j}-M_{i,j})}, \quad j = 1,2,3 \tag{9}$$

and $Y_{i,4}$ denotes the probability of successful interconnection. The following example illustrates the yield estimation process.

Example: Consider that m=4, that is the number of sensed variables is 4, so that the whitening block requires 5 MA_PLUSs, 1 UNL, and 1 DF cells, and that using a 5:1 time multiplexing 1 MA_PLUS, 1 UNL, and 1 DF are needed. Assume that 2 MA_PLUSs, 2 UNLs, and 2 DFs are physically provided. For 16 bit word length and the corresponding area estimates A(MA_PLUS) = 2 mm^2, A(UNL) = 3 mm^2, A(DF) = 2 mm^2, the yield was estimated using the above formula. The results are shown in the top graph of Fig. 7. On the other hand, the *main block* requires 12 MA_PLUSs, 3 UNLs, and 2 DF cells, and that using a 5:1 time multiplexing 3 MA_PLUS, 1 UNL, and 1 DF are needed. Assume that 4 MA_PLUSs, 2 UNLs, and 2 DFs are physically provided. The yield results are shown in the lower graph of Fig. 7.

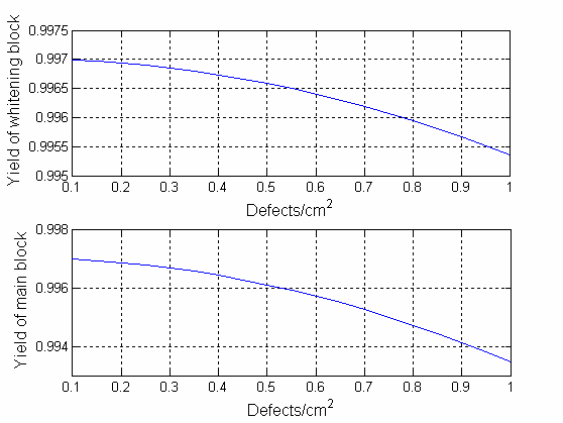

Fig. 7 Estimates of yield for the whitening and main blocks

VIII. Conclusions

We have presented a parallel architecture for the realization of the ICA algorithm using coarse grain cells. Together with time-multiplexing (to be discussed elsewhere in greater detail), this presents the potential for efficiently integrating the ICA with other DSP algorithms, for example event detection or tracking, on a common plane of the 3-D heterogeneous sensor.

References

[1] A. Hyvärinen, J. Karhunen and E. Oja, *Independent Component Analysis,* NY: John Wiley and Sons, 2001.

[2] A. Hyvärinen and E. Oja, "Independent Component Analysis: algorithms and applications", *Neural Networks*, Vol. 13, pp. 411-430, 2000.

[3] S. Bhansali, G. H. Chapman, V. K. Jain, et al. "3D Heterogeneous Sensor System on a Chip", *Proc. SPIE Defense and Security Symposium*, pp. 413-424, April 2004.

[4] G. H. Chapman, V. K. Jain, and S. Bhansali, "Defect Avoidance in a 3-D Heterogeneous Sensor," *Proc. IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 67-75, 2004.

[5] V. K. Jain, S. Shrivastava, D. Damerow, and D. Chester, "Hardware implementation of a nonlinear processor," *IEEE Int. Symp. on Circuits and Systems*, pp. VI-509 to VI-514, May 1999.

[6] V. K. Jain, and L. Lin, "Nonlinear DSP Coprocessor Cell -- One Cycle Chip," *Proc. IEEE Workshop on VLSI Signal Processing*, pp. 256-265, Oct. 1994.

[7] V. K. Jain, and S. Shrivastava, "Rapid system prototyping for high performance reconfigurable computing," *Design Automation for Embedded Systems Jr,* pp. 339-350, August 2000.

[8] V. K. Jain, "Mapping a high-speed wireless communication function to the reconfigurable J-platform," *Proc. IEEE Int. Workshop on Rapid System Prototyping*, pp. 103-108, June 2000.

[9] N. Itoh, et al., "A 600-MHz 54×54-bit multiplier with rectangular-styled Wallace tree", *IEEE Journal of Solid-State Circuits,* pp. 249 - 257, Feb. 2001.