A Highly Dense, Low Power, Programmable Analog Vector-Matrix Multiplier: The FPAA Implementation

Craig R. Schlottmann, *Student Member, IEEE*, and Paul E. Hasler, *Senior Member, IEEE*

***Abstract*—This paper presents a solid foundation for implementing analog vector-matrix multipliers (VMMs) in field-programmable analog arrays (FPAAs). Custom analog VMMs have been demonstrated to be 1000 times more power efficient than commercial digital implementations. However, no previous analog VMM discussion has carefully provided all of the implementation and performance considerations needed to utilize such a system. We utilize the FPAA because it provides an ideal platform for embedding low-power analog processing into larger systems. FPAAs allow the analog processing system to be rapidly prototyped, implemented at low cost, and easily reconfigured in the field. This paper can double as a complete analog VMM design specification, as well as a systematic tutorial on developing general systems with FPAA hardware. We detail the aspects of VMM topology choice, completely analyze the performance metrics, and describe the methods and tools involved in FPAA synthesis.**

***Index Terms*—Embedded systems, hardware/software co-design, low-power design, field-programmable analog array (FPAA) synthesis, formal methods.**

I. Reconfigurable Analog Signal Processing

SIZE, WEIGHT, and POWER (SWaP) are recognized as driving forces in many modern embedded systems. Mobile and tactical systems are often on a fixed power supply, so these forces have a direct effect on the entire performance of the system. Minimizing these costs can result in an increased lifetime, a lighter load for the carrier, or more space for other processing. By attacking the power consumed in traditional embedded processors, we can reduce the rest of the SWaP by allowing for smaller batteries or possibly even moving to harvested energy sources. Although the digital signal processor (DSP) has been making steady gains in power efficiency [1], a true paradigm shift in computing is needed if we hope to make any sizable leap in power efficiency.

This paradigm shift is enabled by signal processors for the analog domain [2], shown in Fig. 1. For instance, when a moderate signal-to-noise ratio (SNR) is allowable, analog processors have recently been demonstrated to be 1000 times more power efficient than comparable digital implementations [3]. This gain in power efficiency can have great implications on the whole field of low-power system design. The core enabler of ultra-low-power operation is the subthreshold transistor. By using currents in the nano-ampere to pico-ampere range, system power can remain in the sub-micro-watt range. Also, analog subthreshold computation can easily be done as a function of gate voltage and does not require any lowering of V_{dd}, which increases integration with other system blocks. One ideal setting for an analog processer is in remote sensor networks. Often placed in difficult or impossible to access areas, remote sensor nodes could remain active for much longer before requiring service or dying. As a compliment to the naturally low current draw, analog processors can operate directly on the incoming signal in its native continuous domain, thus reducing the precision of (or eliminating) the costly conversion stage. In light of the many advantages of analog signal processing, it has been the traditional lack of programmability that has stymied its widespread use in embedded systems.

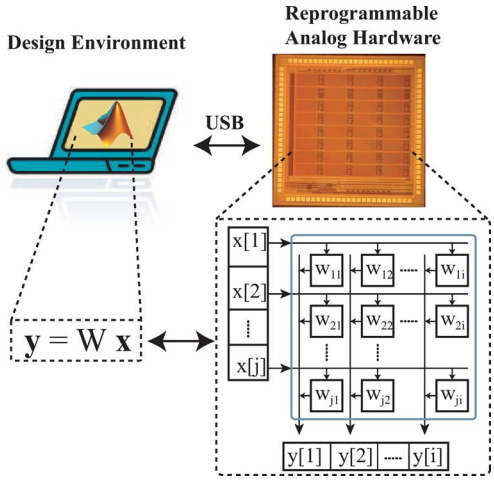


Fig. 1. RASP FPAA design flow. Users can design analog signal processing systems in MATLAB Simulink, then compile that design onto the FPAA hardware. Analog signal processing can provide extreme power efficiency, utilizing subthreshold currents. FPAA design flow facilitates rapid prototyping, fast time to market, and no fabrication costs.

The vector-matrix multiplier (VMM) is a core component in many signal processing applications. Vector-matrix multiplication is commonly performed in FIR filters, 2-D block image transforms, convolution, correlation, and classification [4]. Recently, custom analog VMM cores have provided a low-power, high-throughput tool for signal processing [5]. Several orders of magnitude in efficiency can be gained by allowing the natural physics of the transistors to perform the multiply and accumulate (MAC) operations. Analog VMMs have recently demonstrated low-power solutions in such embedded systems as a transform imager and an OFDM receiver [6], [7]. Although the benefits of analog VMM have been clearly demonstrated, the systematic design of such a system has only been loosely defined in the existing literature.

Manuscript received January 10, 2011; revised June 29, 2011; accepted July 21, 2011. Date of publication October 03, 2011; date of current version November 09, 2011. This paper was recommended by Guest Editor K. Roy.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA. (e-mail:cschlott@gatech.edu; phasler@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2011.2165755

2156-3357/$26.00 © 2011 IEEE

The field-programmable analog array (FPAA) is the ideal tool to incorporate analog signal processing into low-power embedded systems. FPAAs introduce the concept of programmable analog, which brings analog processing in line with what most embedded system designers are comfortable with. This new application space can bridge the gap and allow system-design engineers to unleash the power of analog signal processing. The FPAA gives us a workspace to create a systematic discussion on analog VMM systems. We can use the FPAA to test multiple VMM design choices and put the overall design on solid ground.

The goal of this paper is twofold: first, provide a solid foundation for analog VMM design, and second, formulate a design methodology for reconfigurable analog systems, with the VMM as a case study. In Section II, we provide a brief review of modern FPAAs, the workspace for our analog systems. In Section III, we fully detail the design of analog VMMs. In Section IV, we characterize the performance metrics that are most relevant to analog signal processing. In Section V, we walk through much of the practical matters involved to implement an FPAA-embedded system. Lastly, in Section VI, we provide concluding remarks.

II. Modern FPAA Systems

FPAAs are mixed-signal systems that provide a platform for rapidly implementing analog systems in real hardware. FPAAs consist of a programmable network of switches and routing used to electrically connect analog elements, arranged in configurable analog blocks (CABs). The user of such a device can design, program, and test large scale analog systems in a matter of minutes. This provides the engineer a considerable decrease in fabrication costs and speedup in time-to-market.

Originally reported about twenty years ago [8], FPAAs have seen several architectural revisions. Most have been of modest proof-of-concept size and contained CAB elements that target particular applications. For instance, previous FPAAs have been built specifically as ODE computers [9], or analog filters [10]. More recently it has been the RASP FPAA that has pushed the boundary in both size and performance of what was previously capable [11].

The RASP 2.8a is the largest and most versatile FPAA available [11]. It is composed of 32 CABs, multilevel routing, and on-chip programming structures. The first 28 CABs each contain common analog components: 1 programmable bias operational transconductance amplifier (OTA), 2 programmable input and bias OTAs, 1 OTA buffer, 4 n/pFETs, 2 multiple-input translinear elements, 4 500 fF capacitors, and 1 transmission gate. The other 4 CABs contain elements for signal-by-signal multiplication, such as a Gilbert multiplier. The routing is a full crossbar switch matrix. There are multiple levels of routing lines to reduce the capacitance nodes: local, nearest neighbor, and global.

The core element that facilitates highly dense FPAA systems is the floating-gate (FG) MOS transistor. These transistors have a gate that is electrically isolated from ground, allowing it to store and retain charge. Modern FPAA designs make use of this element as a switch at the intersections of rows and columns in the crossbar switch matrix. No external memory is required

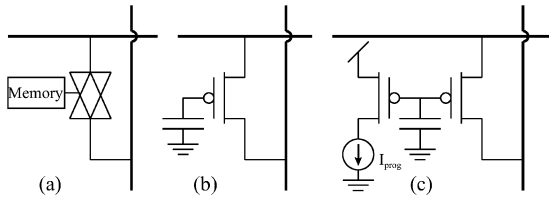


Fig. 2. Three types of switches: (a) traditional switch with a separate memory element; (b) floating gate switch elements store their programmed value, allowing for a very area efficient design; and (c) indirect programming reduces the switches in the signal path.

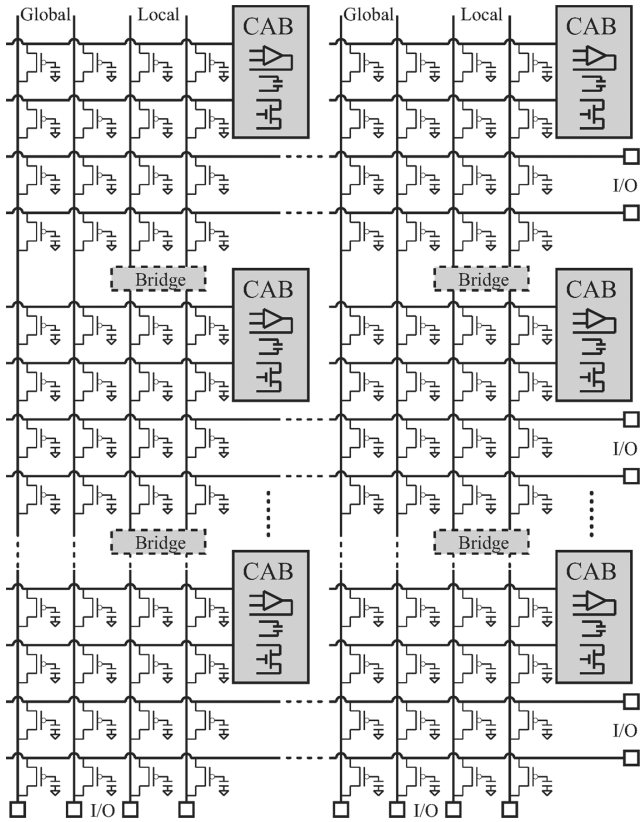


Fig. 3. Architecture of the RASP2.8 FPAA. The routing is full crossbar switch matrix with FG switches. The FG switches can be used for computation, such as for VMM weights.

when utilizing this switch that stores its own value. The FG switch is illustrated in Fig. 2, along with the indirect-programming implementation [12]. The RASP FPAA incorporates all of the necessary circuitry on chip that is required to program the FG switches, thus demanding less overall real estate and supporting structures in embedded systems [13].

One of the most beneficial features of FG switches is that they can also be used as computational elements [14]. FGs can be programmed to hold any value between 'open' and 'closed' essentially providing a free bias source. The use of routing for computation is a major enabler of highly dense FPAA VMMs.

The illustration in Fig. 3 shows the architecture of the RASP 2.8a FPAA and the connections between the CABs, routing, and FG switches. It is fabricated in 350 nm CMOS, with a 2.4 V $V_{\rm dd}$. The RASP 2.8a will be the platform of choice when discussing VMM specifics in the rest of this paper.

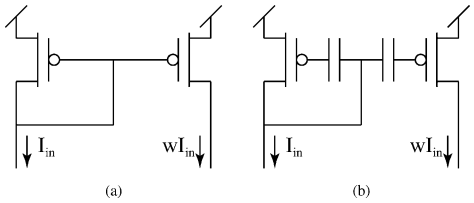


Fig. 4. Current-scaling mirrors. (a) Traditional current mirrors scale current based on the fabricated aspect ratio. Although this can perform a fixed coefficient multiplication, it is not reconfigurable post fabrication. (b) By introducing floating gates into the current mirror, we can set the scaling weight based on the amount of charge on the floating node.

III. Building VMM on FPAA Devices

In this section, we provide a thorough discussion on the design of analog VMMs. Although the RASP FPAA is versatile enough to implement any topology discussed here, we will guide our design choices towards the most efficient use of the FPAA architecture.

A. Analog Vector-Matrix Multiplication

Vector-matrix multiplication is mathematically defined as $\vec{y} = W\vec{x}$ where $W \in \mathbb{R}^{m\times n}$, $\vec{x} \in \mathbb{R}^n$, and $\vec{y} \in \mathbb{R}^m$. We restrict our discussion to real values, since we will be dealing with physical quantities. To get a sense of the analog elements needed to perform this task, we look at the component-wise output signal:

$$y_i = \sum_{j=1}^{n} w_{ij}x_j,\ i = 1,\ldots,m. \tag{1}$$

Each element of the output vector is made up of the inner product of the input vector with a row of the matrix, an operation requiring scalar multiplication and addition. This component-wise operation is shown in the breakout of Fig. 1.

Fortunately, by utilizing current-mode signals, scaling and adding are two very easy and efficient operations. Addition can be performed by Kirchoff's current law (KCL), by mixing two currents. This function requires no power to perform. Scaling currents is simple to perform as well. One transistor is used to sense the input current and broadcast the log-compressed voltage, while another transistor receives the voltage and exponentiates it back into a current. This operation is recognizable as a current mirror, illustrated in Fig. 4(a). A common CMOS current mirror scales current based on its geometry: $I_{\text{out}} = I_{\text{in}}(W/L)_2/(W/L)_1$. This scaling is commonly used to create bias currents as multiples of a reference current.

Although the common-current-mirror approach performs our desired scaling, our application demands a system that is capable of being rescaled after fabrication. Fig. 4(b) illustrates the use of a FG mirror to create a programmable scaling value [15]. To achieve ultra low power, we operate each transistor in the subthreshold regime, where the drain current has the following exponential dependence on gate voltage:

$$I_d = I_0 e^{\frac{V_s - \kappa V_g}{U_T}}. \tag{2}$$

Here, I_0 is a pre-exponential constant term, κ is the capacitive division between the oxide capacitance and the depletion capacitance, and U_T is the thermal voltage. For this pFET, all potentials are referenced to the bulk. Analog subthreshold operation is performed by using gate biases below the threshold value. This facilitates integration with other systems, because we can leverage the subthreshold current levels without the low V_{dd} values that typify subthreshold digital design.

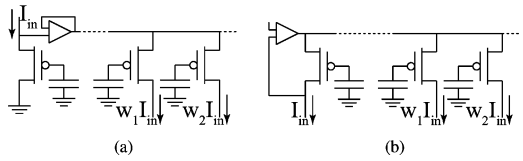


Fig. 5. Two implementations of floating-gate source-coupled current mirrors. (a) Buffered input stage: the source voltage is buffered with an OTA follower. (b) Log-amp input stage: a logarithmic amplifier is used to compress the input current.

The mirror's scaling factor is found by taking the ratio of the output to the input current,

$$\frac{I_{D,\text{out}}}{I_{D,\text{in}}} = e^{\frac{\kappa(V_{fg,\text{out}} - V_{fg,\text{in}})}{U_T}} \equiv w. \tag{3}$$

Here, V_{fg} is the voltage on the floating node, which is a function of the stored charge and any capacitively coupled voltages.

Although the gate coupling of Fig. 4 serves the purpose of a weighted current mirror, we can also use the source-coupled topology of Fig. 5. Here too, the input current is sensed, log compressed, and broadcast. This topology allows us to take advantage of the many thousands of FG switch elements in modern FPAAs, which are two-terminal devices. Source coupling is also beneficial because it eliminates the effect of kappa variation with input signal. However, whereas the gate is a high-impedance node and has no current draw, the source will sink current, so it must be broadcast. Fig. 5 shows two ways of broadcasting the source voltage, referred to in this paper as buffered and log-amp stages, respectively. Source coupling involves forcing the input current into the source of the sensing FET, then buffering the source voltage to the output stages. The bandwidth of the buffered structure is $\omega = g_m/C$, where $g_m = I\kappa/U_T$. The log-amp structure uses the amplifier of [16] as the sensing stage. This incorporates active feedback which increases the bandwidth to $\omega = Ag_m/C$, where A is the voltage gain of the OTA, which is typically 100–200 on the RASP FPAA.

As mentioned in the previous discussion, the architecture of the FPAA guides the design of the output array. As defined in (3), each output weight depends on the ratio of the output FG charge with the input state's charge. We achieve this ratio by connecting an FG element's source to the broadcast line. For the summing operation we get a current-mode addition by tying the drains of multiple output elements together. This fits exactly with what is available in the RASP FPAA: an array of FG switches that are source coupled along a row and drain coupled along a column.

Pulling all of this together, we have the structure in Fig. 6. We use the log-amp structure to log compress each of the n elements in the input vector, broadcasting the resultant voltage across a row. Each row has m output-stage FGs, creating the multiplier weights. The output current of each scaling element is then summed along the m columns. With FG transistors, the output impedance is mainly degraded by the drain voltage coupling back onto the floating node. Adding a cascode transistor

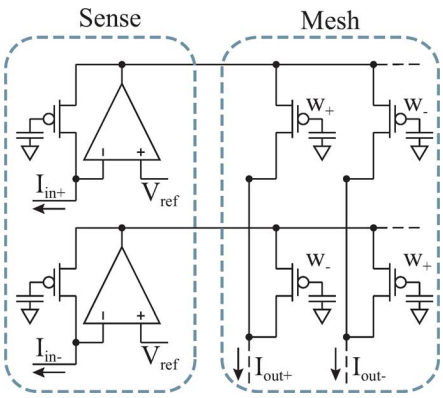


Fig. 6. Schematic of the VMM. The input floating-gate stage produces a log-compressed voltage representation of the input current. This voltage is broadcast to the source of each FG element in that row. These FGs produce an output current that is a scalar multiple of the input current, which are then summed along a column by KCL.

at the output of each column helps to reduce the effect of the drain voltage on the computation. Thus, we have successfully implemented (1) in an extremely compact, highly dense fashion on the RASP FPAA.

B. Signal Conditioning

From the structure in Fig. 6, it is clear that the input and output currents must be unidirectional. This results in weights that need to be strictly positive. In this scenario, we are left with a single-quadrant multiplier: positive signals and weights.

Ideally, we would prefer to have full four-quadrant multiplication: both positive and negative signals and weights. To achieve this, we incorporate a differential syntax. We define the signed signal to be the difference between two positive currents:

$$I_{\text{in}} = I_{\text{in}+} - I_{\text{in}-}. \tag{4}$$

We will constrain the differential signals to small changes around a bias (I_B) current

$$\frac{I_{\text{in}+} + I_{\text{in}-}}{2} = I_B. \tag{5}$$

Now, we will utilize a similar syntax for the weights

$$w_+ = w_B + \frac{\Delta w}{2},\ w_- = w_B - \frac{\Delta w}{2}. \tag{6}$$

For one four-quadrant differential multiplier element, we have

$$\begin{bmatrix} w_+ & w_- \\ w_- & w_+ \end{bmatrix} \begin{bmatrix} I_{\text{in}+} \\ I_{\text{in}-} \end{bmatrix} = \begin{bmatrix} I_{\text{out}+} \\ I_{\text{out}-} \end{bmatrix}, \tag{7}$$

which is shown in Fig. 6. This core multiplier cell has a mesh four times as large as the single mirror. The final gain of the four-quadrant cell is

$$I_{\text{out}+} - I_{\text{out}-} = (I_{\text{in}+} - I_{\text{in}-}) \cdot (w_+ - w_-). \tag{8}$$

The differential-signal operation is illustrated in Fig. 7 [17]. An additional benefit of differential signals is that it will remove DC offsets and even-order harmonics.

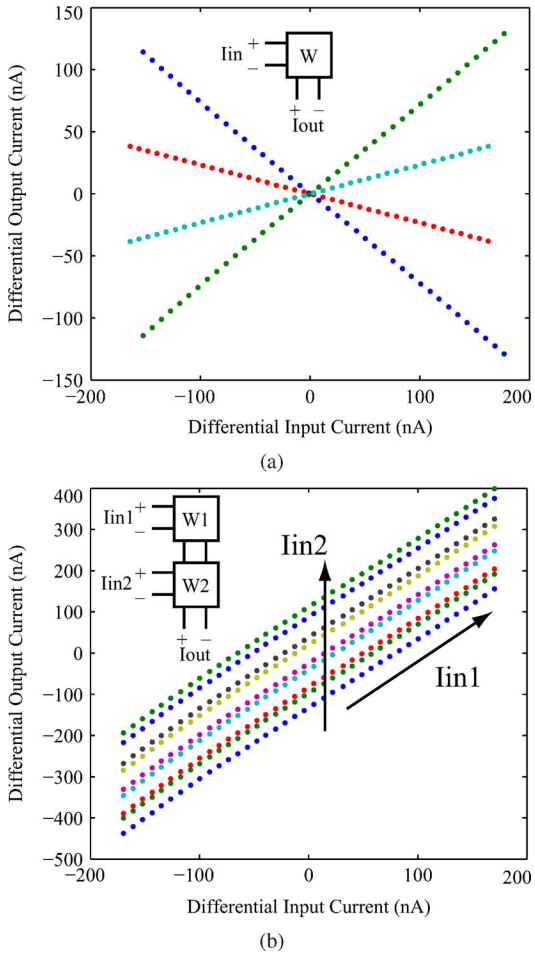


Fig. 7. VMM characterization. (a) 1×1 VMM; the inputs were swept differentially, to create differential outputs. The gains for this sweep were ≈ ±1, ±0.5. (b) 2×1 VMM; this sweep demonstrates the summation of the two inputs. One input is swept differentially for each constant value of the second input, resulting in vertical offsets.

IV. Power, Speed, Noise, and Temperature Dependence

In this section, we discuss the relevant performance parameters. The power-delay product illustrates an important trade-off in subthreshold design: speed at the expense of power. Also, two of the most often criticized shortcomings of analog computation are discussed: noise and temperature dependence. Although these effects are unavoidable, by characterizing them we can understand their effects and design the rest of the system to compensate for them.

A. Power and Speed

One of the most attractive features of subthreshold processing is its extreme power efficiency. This efficiency is very important for VMM applications in mobile devices or any system on a limited power budget. However, this low-power operation comes at the cost of operating speed, manifest in the *power-delay product*. This discussion is in similar fashion to that given in [18].

The input stage is shown to dominate the frequency response of the VMM system. The small-signal analysis of the input stage follows the analysis given for the log-amp [16]. For the case

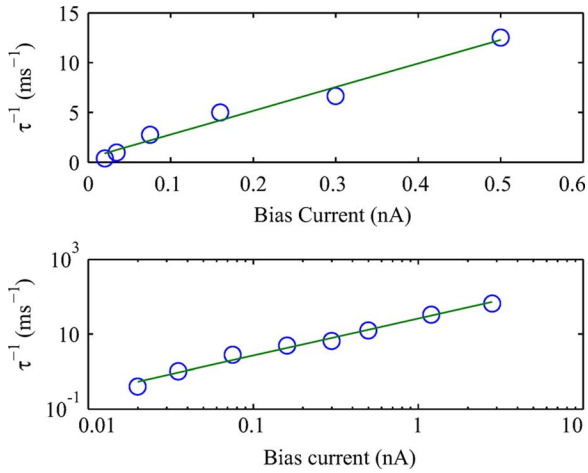


Fig. 8. Linear- and log-plot of the inverse VMM time constant. The inverse of the time constant increases linearly with an increase in bias current. The slope of the linear plot corresponds to an input capacitance of 1.62 pF. The slope of the log plot is 0.99.

where the input capacitance is much larger then the parasitic feedback capacitance, the dominant pole is given as:

$$p = -\frac{AG_{s1}}{C_{\text{in}}} \tag{9}$$

where G_{s1} is the source conductance of the feedback FET, A is the voltage gain of the OTA, and C_{in} is the total capacitance at the input.

Here, the benefit of the OTA can be seen for the log-amp input stage; the input impedance is decreased from buffered input stage by the factor A, increasing the bandwidth by that same amount. The typical value of A on an OTA of the RASP FPAA is 100–200, so a sizable increase in bandwidth can be achieved.

By substituting the subthreshold equation for G_s we see a -3 dB frequency of

$$f_{-3\text{ dB}} = \frac{1}{2\pi}\frac{AI}{C_{\text{in}}U_T}. \tag{10}$$

This shows that the operating frequency scales linearly with signal bias, and thus power. Fig. 8 shows the speed of the response of the VMM for given bias currents, shown both on linear and log plots. The slope of the linear plot empirically gives us an input capacitance value of 1.62 pF, a reasonable result given the reconfigurable nature of the routing. The slope of the log plot is 0.99, which is expected from the linear dependence.

This results in a power-delay product for our MAC cell of

$$P\tau = 6V_{\text{dd}}U_T\frac{C_{\text{in}}}{A}. \tag{11}$$

The power is approximated as the product of the total current and the supply voltage. The factor of 6 is due to the power consumed in the OTA of a 1×1 cell; it provides current to supply both the input and output stages, and it has a copy of the total current in each of 3 internal branches. The full power will scale with the size of the matrix: $I = 3 \cdot \text{row} \cdot (1 + \text{col})$. This power-delay equation is shown to be a linear function of capacitance and independent of the signal bias. In this log-amp input stage, the input capacitor has effectively been reduced by the factor A. The inverse of this product can also roughly be considered the computation per unit power.

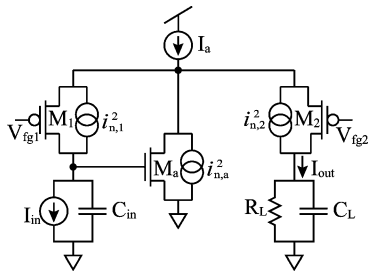


Fig. 9. Schematic of the 1×1 VMM noise model.

B. Noise Performance

On the topic of analog computation, the issue of noise performance is very important. We will analyze the core 1×1 cell of the log-amp source-coupled VMM following the discussion in [16]. To create an equivalent noise model, shown in Fig. 9, we consider channel noise current sources for each transistor of the FG mirror and the OTA. Since we are restricting our operation to subthreshold currents, we will neglect flicker noise and focus on thermal noise.

The thermal noise contributions are

$$\hat{i}^2_{\text{out}} = \tilde{i}^2_1\frac{g^2_{s2}}{g^2_{s1}} + \tilde{i}^2_a\frac{g^2_{s2}}{g^2_{ma}} + \tilde{i}^2_2. \tag{12}$$

Referring the noise to the input, using the noise model $\hat{i}^2 = 2qI\Delta f$ [19], and substituting the subthreshold transconductance, we get

$$\hat{i}^2_{\text{in}} = 2qI_1\left(1 + \frac{I_1}{\kappa I_a} + \frac{I_1}{I_2}\right)\Delta f. \tag{13}$$

We use the bandwidth found in (10) and utilize the relation that the ratio $I_2/I_1 = w$. We note that the amplifier bias needs to source current for each FG of the current mirror, initiating the constraint $I_a \geq I_1(1 + w)$. Given these definitions, (13) becomes

$$\hat{i}^2_{\text{in}} = 2qI_1^2\left(1 + \frac{1}{\kappa(1+w)} + \frac{1}{w}\right)\frac{A}{4C_{\text{in}}U_T}. \tag{14}$$

What this highlights is the increase in noise power with A, effectively decreasing C_{in}. This goes back to the trade-off in topologies chosen for higher bandwidth. For lower noise, we can use a topology without the amplifier or increase the input capacitance. Fig. 10 shows a plot of the current spectral density taken from a VMM compiled on the RASP 2.8a FPAA.

For the current mirror, the input signal is also the bias of the input stage. We will constrain the VMM operation to weights of a small range around 1, in which case we will use the average $w = 1$ for the final signal-to-noise (SNR) relation. With $w = 1$ and $\kappa \approx 0.5$, the three terms summed in the parenthesis can each be approximated as unity. The RMS SNR is now in the relation:

$$\text{SNR} = \sqrt{\frac{2C_{\text{in}}U_T}{3Aq}}. \tag{15}$$

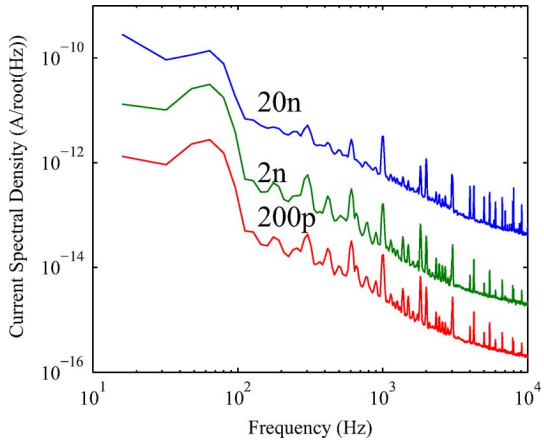


Fig. 10. Current spectral density of the VMM. The lower bias current produces less current noise.

We see from this equation that this multiplier's SNR is not dependent on input bias, but is increased by the input capacitor. However, the performance trade-off with increasing the input capacitance is an increase in power-delay. This trade-off is illustrated in the following relation:

$$\frac{P\tau}{\mathrm{SNR}^2} = 72qV_{\mathrm{dd}}. \tag{16}$$

C. Temperature

A weight produced by subthreshold-current ratios will see a temperature dependence. This is unavoidable, but we can characterize it and thus compensate for it. To determine the temperature dependence, we will rewrite the weight from (3) in terms of a constant thermal voltage, $U_T = U_{To}T/T_o$

$$w = w_o^{\frac{T_o}{T}}. \tag{17}$$

The actual temperature is a change around a constant temperature: $T = T_o + \Delta T$. Now, lets say $\Delta T/T_o \ll 1$, so $T_o/(T_o + \Delta T) = 1 - \Delta T/T_o$. Using the relation from (8), we have the VMM differential weight in temperature form

$$\frac{I_{\mathrm{out}+} - I_{\mathrm{out}-}}{I_{\mathrm{in}+} - I_{\mathrm{in}-}} = w_{1o}^{\frac{T_o}{T}} - w_{2o}^{\frac{T_o}{T}}. \tag{18}$$

Imposing the constraints from (16), we get

$$\Delta w = \left(w_{Bo} + \frac{\Delta w_o}{2}\right)^{1-\frac{\Delta T}{T_o}} - \left(w_{Bo} - \frac{\Delta w_o}{2}\right)^{1-\frac{\Delta T}{T_o}}. \tag{19}$$

Using the binomial theorem, we can expand the terms

$$\Delta w = \left(1 - \frac{\Delta T}{T_o}\right)(\Delta w_o) + \frac{1}{24}\left[\frac{\Delta T}{T_o} - \left(\frac{\Delta T}{T_o}\right)^3\right]\Delta w_o^3. \tag{20}$$

We benefit from the differential structure by seeing the even-order terms drop out. From this, we can drop the higher-order terms based on our original constraint that $\Delta w < 1$. We are left with a weight that is linear in the inverse of temperature change

$$\Delta w = \left(\frac{T_o}{T}\right)(\Delta w_o) + O\left(\Delta w_o^3\right) \tag{21}$$

To verify our assumptions when deriving this simplified model, we have plotted (19)–(21) in Fig. 11(a). A comparison is shown in Fig. 11(b) of the weight versus temperature for the differential- and single-ended cases. The differential case is much less drastically affected by the change in temperature.

With the temperature dependence now defined, we can design the surrounding structures to compensate for this dependence. One common way to introduce a temperature term in the numerator is to pass the current-mode signal through a diode-connected nFET. The output voltage signal is now

$$V_{\mathrm{out}} = \frac{U_T}{\kappa}\ln\left(1 + \frac{\Delta I_{\mathrm{out}}}{I_{\mathrm{bias}}}\right). \tag{22}$$

By taking the signals differentially and expanding the log terms, (22) is:

$$V_{\mathrm{out}}^+ - V_{\mathrm{out}}^- = \frac{U_T}{\kappa}\left[\frac{U_{To}}{U_T}\Delta w_o\left(I_{\mathrm{in}}^+ - I_{\mathrm{in}}^-\right)\right] \tag{23}$$

canceling the temperature dependent term.

On the input side, we can introduce a temperature-neutral V-to-I stage with a differential (diff) pair. For the diff pair to be temperature neutral, it should have a proportional-to-absolute-temperature (PTAT) current source. This overall system is shown in Fig. 12. The VMM is now in voltage mode which makes it much easier to interface with other elements in an embedded system.

Table I shows a compilation of the performance metrics discussed in this section. The parameters are calculated for three common values of signal bias, based on a single differential cell (2×2). In the calculations, we used the input capacitance found in the delay measurements and w of 1. The computation per power ($\mathrm{MMAC}/\mu\mathrm{W}$) is roughly the inverse of the power-delay product in (16). The computation per power and SNR are shown to be constant and independent of bias. The 68 nW for 1 million MAC operations used by this source-coupled architecture is a marked improvement in power efficiency over the 270 μW in the custom analog VMM in [5], which was itself a 1000 times improvement over commercially available DSPs.

V. Methods and Tools for FPAA VMMs

In this section, we discuss the practical matter of implementing a VMM into an embedded system. We provide a full discussion on the density issues when utilizing an FPAA. To make it easy for engineers to utilize analog VMMs, we have incorporated it into the software compiler tools. In addition, we discuss a few supporting blocks, also supported by the software tools, that make designing full systems very easy.

A. FPAA Density

The architecture of the RASP FPAA is particularly well suited to implement the VMM structure discussed in Section III. By utilizing the FG switches for computation, we can effectively use the switch matrix as the VMM mesh. Fig. 13 shows an illustration of how the core differential 1×1 MAC block can be compiled into the switch matrix.

When discussing VMM density, the question arises of how large of a VMM can be built. Highly parallel processing is where the most computational gains can be made with analog. The

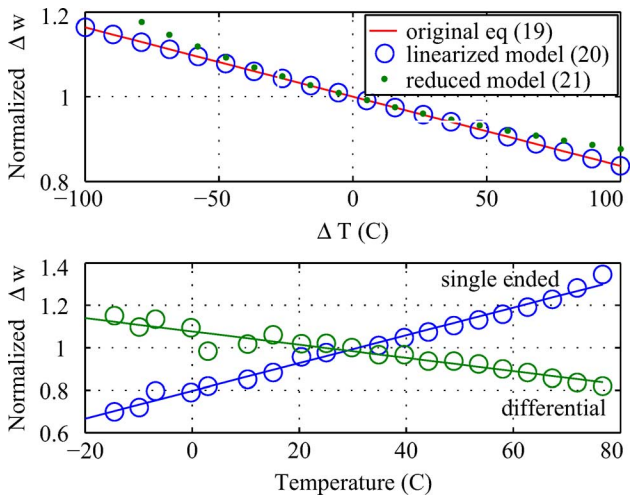


Fig. 11. Temperature dependence of the VMM weight. (a) Plotting the derived equations shows that we were founded in our assumptions; the linearized model very closely follows the original equation and the reduced model fits for ΔT within ± 50 C. The weights are normalized to the programming condition (30 C) (b) Temperature sweep of the weights from the FPAA VMM FPAA. The differential weights are shown to have a much smaller slope than the single-ended multiplier. The weights are normalized to the programming condition (30 C).

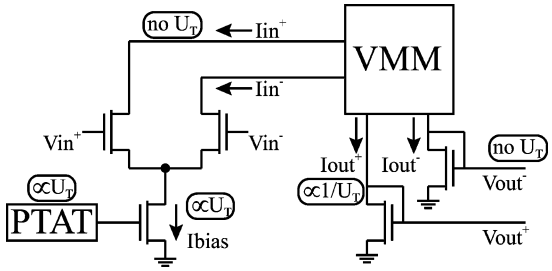


Fig. 12. Schematic of the temperature compensation circuitry. Diode-connected nFETs at the output cancel the temperature dependence of the multiplier weight. The input diff-pair with PTAT current source will have no temperature dependence.

TABLE I
SUMMARY OF PERFORMANCE PARAMETERS FOR 1×1 DIFFERENTIAL CELL. WE USE: ROWS (R) AND COLUMNS (C) $= 2$, $C_{\text{in}} = 1.6$ PF, $V_{\text{dd}} = 2.4$ V, $A = 165$, AND $w = 1$

Property	Expression	$100pA$	$1nA$	$10nA$
Bandwidth (f)	$\frac{AI}{2\pi C_{in} U_T}$	63kHz	630kHz	6.3MHz
Power (P)	$3r\,(1+c)\,IV_{dd}$	4.3nW	43nW	430nW
Noise $\left(\hat{i}_{out}\right)$	$\sqrt{\frac{3qI^2A}{2U_T C_{in}}}$	3.1pA	31pA	310pA
$MMAC/\mu W$	$\frac{A}{36\pi V_{dd} U_T C_{in}}$	14.6	14.6	14.6
SNR	$10log_{10}\left(\frac{2U_T C_{in}}{3qA}\right)$	30.2dB	30.2dB	30.2dB

RASP 2.8a is one of the most recent and advanced FPAAs, with 55 general I/O and a 16-bit scanner I/O block; however, newer and larger RASP FPAAs are continuously being developed for which the same analysis will hold.

When using an entire FPAA solely for a VMM, the first constraint is the analog I/O limit. With 55 I/O, the largest square computation would be 27×27. Of course, the matrix by no means has to be square. In addition, the on-chip scanner can be used to multiplex multiple results to a single I/O pin. With the scanner, a 55×16 VMM is possible, with the 16 outputs being available in series.

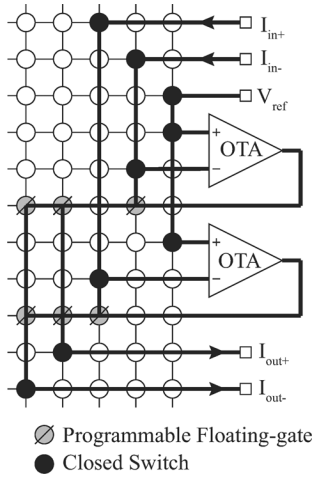


Fig. 13. Map of the 2×2 VMM implemented on the FPAA. VMM utilizes floating gates programmed as both switches and computational elements. This allows very efficient utilization of all chip area and results in highly dense multipliers.

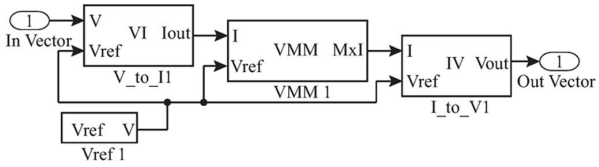
The discussion gets more interesting if dealing with a VMM internal to a system on the FPAA, where the signals do not need to be pinned out; in this case, the density is routing-limited. The RASP 2.8a is symmetric with 4 identical columns of CABs (cabstacks) which can be treated independently in the calculation and summed down the rows. Each cabstack contains 21 OTAs and 32 vertical lines. By referencing the diagram in Fig. 13, we see that each input needs a vertical line and an OTA, each output needs a vertical line, and the reference voltage needs one vertical line. Each cabstack VMM is constrained by: inputs + outputs ≤ 31, with the inputs ≤ 28. The total VMM is then four times this number by accounting for the four cabstacks. To put in numbers, one possible VMM is calculated to be 82×10.

B. Compiler Tools

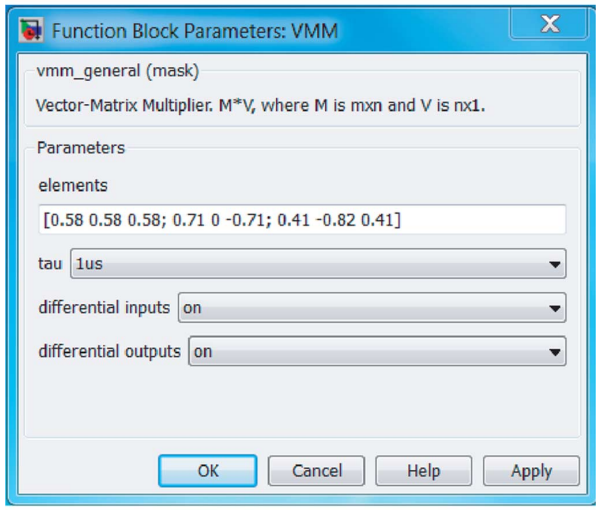
To make it as easy as possible to design FPAA systems with the VMM block, we have incorporated it into the Simulink compiling tool, Sim2spice. This tool allows engineers to design analog signal processing systems at the block level in Simulink, then compile that design onto the FPAA [20].

By adding a block to the Simulink library, the main objective is to abstract the design to as simple as allowable. The Simulink tool compiles down to a netlist, so full transistor-level simulation can be done in SPICE. The Simulink model will capture the important signal attributes without bogging down the simulation time. Certain design parameters are abstracted and presented to the user in a fashion that is intuitive. Fig. 14 shows the block design of a system in Simulink using the VMM and the GUI dialog box which corresponds to the VMM. A graphical interface (the RAT) is provided to visualize the compiled chip utilization, shown in Fig. 15.

Once the VMM design is compiled from Simulink, it can be programmed and tested using our RASP Program & Evaluation (RPE) board [21]. This platform allows us to fully test the system before embedding it into a larger system. The RPE board communicates with MATLAB on a PC via a USB connection. The board provides 40 DAC channels and 12 ADC channels for test-benching systems.



(a)



(b)

Fig. 14. (a) Simulink block level design for a VMM system, including supporting blocks. The Sim2spice tool is used to compile this block diagram down to object code ready to be programmed on the FPAA. (b) The Sim2spice parameters for the VMM block. Rather than overwhelm the user with circuit specifics, the functionality has been abstracted for high-level system design.

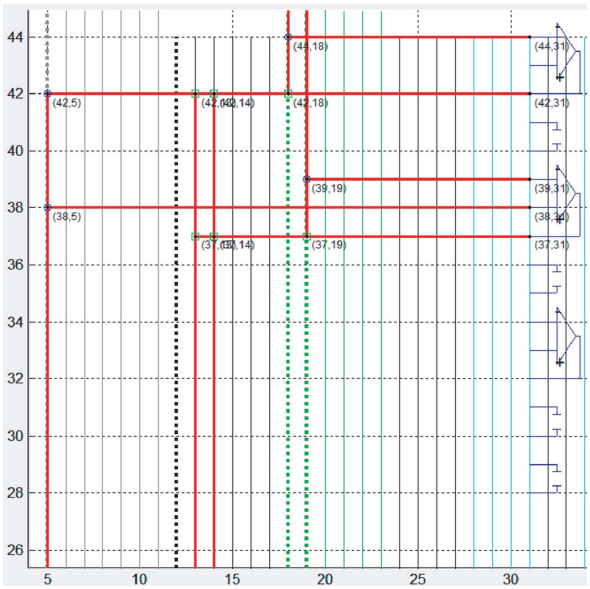


Fig. 15. The RAT visualization tool. This tool allows the user to view and modify the switches that will be programmed on the FPAA.

C. Supporting Blocks

In addition to the core VMM block, it is helpful to have several supporting blocks available in the design library. Since the VMM is a current-in/current-out system, we have created several signal conversion blocks to transform the input/output into the voltage domain.

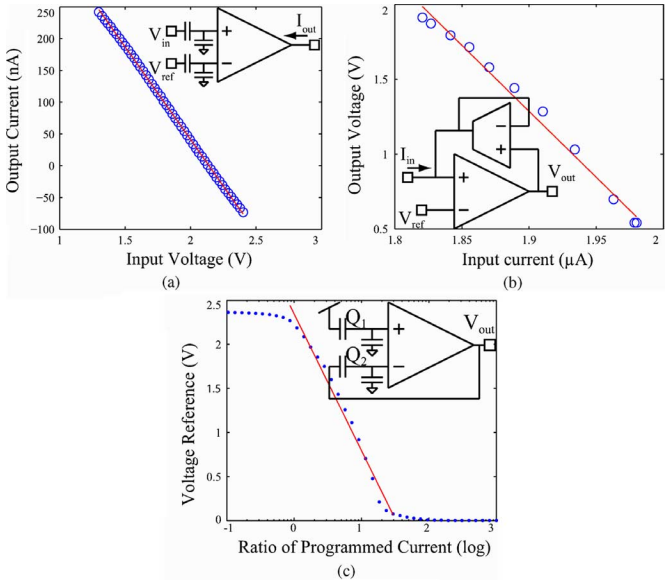


Fig. 16. Supporting blocks. (a) Input stage V–I converter, (b) output stage I–V converter, and (c) programmable voltage reference. The design for these blocks was highly motivated by the available components in the FPAA, in this case OTAs.

The V/I block is shown in Fig. 16(a). The design for this block was highly motivated by the CAB elements in modern FPAAs; this particular design only needs 1 wide-linear-range OTA. The expanded linear range is a result of the capacitive attenuation on the input. Other V/I converters are certainly possible, such as the differential pair discussed in Section IV-C which is good for designing for temperature.

The I/V block is shown in Fig. 16(b). Again, this design was highly motivated by the CAB elements, highly utilizing OTAs. Here we display the output characteristics of a transimpedance amplifier (TIA). The TIA has the benefit of being able to convert bi-directional currents, although this means that the two differential output currents of the VMM must first be combined with a current mirror.

As a final example of Simulink supporting blocks, the voltage reference is shown in Fig. 16(c). This is implemented with a single FG-input OTA with negative feedback. By programming a different ratio of currents on the inputs, we can effectively create a fixed output voltage on the OTA. This block is useful for setting all of the references, without burning an I/O pin, which are valuable for large dimension VMM computation.

VI. Conclusion

We have presented a thorough design methodology for implementing a vector-matrix multiplier on a field-programmable analog array. We first highlighted the power of analog signal processing and FPAAs in particular. FPAAs provide an ideal framework for ultra low power in embedded system design. We next described the structure of the analog VMM and discussed some of the topology trade-offs. The trade-offs became clear in the analysis section, in terms of speed, noise, and temperature performance. The topology that we focused on proved to have extremely high computational power efficiency. Lastly, we elaborated on the practical implementation. There is an entire tool-set infrastructure available for designing analog systems in