ON LOW-POWER ANALOG IMPLEMENTATION OF PARTICLE FILTERS FOR TARGET TRACKING

Rajbabu Velmurugan, Shyam Subramanian, Volkan Cevher, David Abramson, Kofi M. Odame, Jordan D. Gray, Haw-Jing Lo, James H. McClellan, and David V. Anderson

Georgia Institute of Technology, Atlanta, GA 30332-0250, USA

ABSTRACT

We propose a low-power, analog and mixed-mode, implementation of particle filters. Low-power analog implementation of nonlinear functions such as exponential and arctangent functions is done using multiple-input translinear element (MITE) networks. These nonlinear functions are used to calculate the probability densities in the particle filter. A bearings-only tracking problem is simulated to present the proposed low-power implementation of the particle filter algorithm.

1. INTRODUCTION

Particle filters [1] are used in state estimation, where the underlying state-space models can be nonlinear and non-Gaussian. They use a large number of weighted samples, called particles, to represent probability distributions involved in the estimation. Because particle filters do not approximate the nonlinearities in the state-space systems, they are computationally complex. Hence, some particle filter applications require real-time hardware implementations that must be efficient in speed, accuracy, and power consumed.

Advances in the field of semiconductor technology and communications have led to fast and efficient distributed signal processing algorithms in a network of sensor nodes. In most applications, these sensor nodes operate under constrained energy and power. Hence energy consumed in communication and computation is critical and needs to be minimized. A promising approach to achieve low power is to have cooperative systems that have both digital and analog components [2]. This will be the approach adopted in this work.

In this paper, we propose a low-power analog implementation of particle filters. The analog implementation uses Multiple-Input Translinear Element (MITE) networks to perform certain nonlinear functions. MITEs were introduced as a circuit primitive in [3]. The proposed analog implementation is demonstrated in the particle weighting stage of a bearings-only target tracker. We compare the power consumed at the weight evaluation stage of the proposed analog implementation with that of a digital implementation. Specifically, the power consumed in *arctan* and *Gaussian* function evaluations is given. Most digital implementations of these functions use the COordinate Rotation DIgital Computer (CORDIC) algorithm [21]. These functions also determine the update rate of the particle filter algorithm [4].

Hardware implementation of particle filters was also addressed in [4]. An architecture was presented for efficient digital hardware implementation of particle filters along with an efficient implementation of the resampling stage. This resulted in a Field Programmable Gate Array (FPGA) prototype for a particle filter algorithm. This was followed by VLSI or Application Specific Integrated Chip (ASIC) development for particle filters [5], [6]. Other works addressed the issue of mitigating the complexity of particle filters by proposing structural modifications such as efficient pipelining and parallel operation [7], [8].

The remainder of this paper is organized as follows. Section 2 describes the particle filter based tracking considered in the implementation. In Sec. 3, MITEs are introduced and implementation of non-linear functions using MITEs is presented. Section 4 describes the proposed particle filter implementation strategy and also presents a comparison of analog and digital power consumption. Simulation results are presented in Sec. 5, with discussions, future work in Sec. 6, and conclusions in Sec. 7.

Prepared through collaborative participation in the Advanced Sensors Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-02-0008. This research was partially supported by Texas Instruments. We also thank Sunil Kamath at Georgia Tech for the useful discussions.

2. PARTICLE FILTER BASED TRACKING

2.1 Particle filter

Particle filters or sequential Monte Carlo (SMC) methods are a class of recursive simulation methods for solving filtering problems [1], [9]. Consider a state-space system where $\mathbf{x}_k$ is the state vector and z_k are the noisy measurements related to the state at time k. We are interested in estimating the state $\mathbf{x}_n$ at time n given the measurements z_k for $k = 1, \ldots, n$. The state transition density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and data likelihood $p(z_k|\mathbf{x}_k)$ can be obtained from the state transition and measurement equations. A Bayesian solution for this filtering problem involves obtaining the posterior distribution

$$\pi_n = p(\mathbf{x}_n|z_n, \mathbf{x}_{n-1}) \propto p(z_n|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{x}_{n-1}). \tag{1}$$

Particle filters use a combination of importance sampling, weight update and resampling to sequentially obtain and update the distribution (1). Importance sampling approximates a probability distribution π using a set of N weighted particles $\{\mathbf{x}^{(i)}, w^{(i)}\}$, where $\mathbf{x}$ represents the state and w the corresponding weight for the i^{th} particle. The weight update stage uses measurements to update particle weights. The updated particles are used to make inferences on the state $\mathbf{x}_n$. The resampling stage avoids degeneracy by removing particles with low weights and replicating particles with high weights.

2.2 Bearings-only tracking

Bearings-only tracking involves estimating the target states based on angle measurements at a sensor node. The target is assumed to move in the *x*-*y* plane and to follow a constant-velocity motion model [10], with a state update period of 1s. The state transition is described using the relation

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{\Gamma}\mathbf{u}_k, \tag{2}$$

where $\mathbf{x}_k = [x \quad v_x \quad y \quad v_y]_k^T$, $\mathbf{u}_k = [u_x \quad u_y]_k^T$,

$$\mathbf{F} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } \mathbf{\Gamma} = \begin{pmatrix} 0.5 & 0 \\ 1 & 0 \\ 0 & 0.5 \\ 0 & 1 \end{pmatrix}. \tag{3}$$

Here, x and y are the Cartesian coordinates of the target, v_x and v_y are the corresponding velocities. Parameter $\mathbf{u}_k$ represents the system noise and is Gaussian distributed with covariance $\mathbf{\Sigma}_u = \sigma_u^2\mathbf{I}_2$, where $\mathbf{I}_2$ is a 2 x 2 identity matrix.

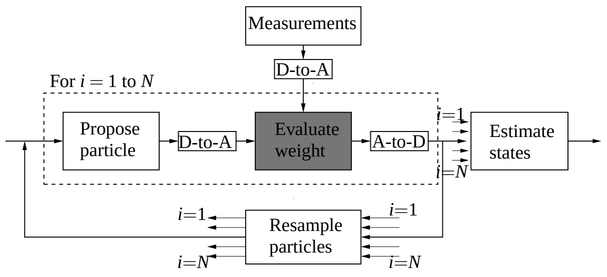

Figure 1: Block diagram showing computational flow in the particle filter algorithm. The flow shown here is for the implementation Method-1 in Sec.4.1. Highlighted stage is performed in the analog domain.

Table 1: Bearings-only tracker: Particle filter pseudocode

```
Given the observed data $z_k$ at $k$,
1. For $i = 1,2,\ldots,N$ sample or propose particles,
   $\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})$ using (2).
2. For $i = 1,2,\ldots,N$ calculate the weights,
   $\hat{w}_k^{(i)} = w_{k-1}^{(i)} p(z_k|\mathbf{x}_k^{(i)})$,   (5)
   where $p(z_k|\mathbf{x}_k^{(i)})$ is the observation density, given by
   $\frac{1}{\sqrt{2\pi\sigma_r{}^2}} \exp - \frac{\left(z_k - \arctan \frac{y_k^{(i)}}{x_k^{(i)}}\right)^2}{2\sigma_r{}^2}$.   (6)
3. Calculate normalized weights $w_k^{(i)}$ using $\hat{w}_k^{(i)}$.
4. Make estimates, $E\{\mathbf{x}_k\} = \sum_{i=1}^N w_k^{(i)}\mathbf{x}_k^{(i)}$.
5. Resample $\left\{\mathbf{x}_k^{(i)}, w_k^{(i)}\right\}$ to obtain new set of particles
   $\left\{\mathbf{x}_k^{(j)}, w_k^{(j)} = \frac{1}{N}\right\}$.
```

The angle measurements at a stationary sensor are given by

$$z_k = \arctan\{y_k/x_k\} + r_k, \tag{4}$$

where r_k represents a Gaussian measurement noise with mean zero and variance σ_r.

Using (2) and (4), a particle filter algorithm for target tracking similar to the one in [9] can be formulated. The state update is used to propose new particles. This provides a sub-optimal recursive estimate of the target position in the *x*-*y* plane. The pseudocode for the particle filter algorithm is shown in Table 1 and a block diagram of the computational flow is shown in Fig. 1. For comparison, a digital hardware implementation of this algorithm is presented in [4].

3. ANALOG IMPLEMENTATION

3.1 Multiple-Input Translinear Element (MITE)

The multiple-input translinear element (MITE) was introduced in [3] as a generalization of the bipolar transistor. Both these circuit elements use their exponential transfer characteristic to implement nonlinear functions or systems [11], [12]. The system parameters are represented by currents and are hence tunable. The only MITEs relevant to this paper are 2-input MITEs. A 2-input MITE, whose symbol is shown in Fig. 2(a), is defined as a circuit element satisfying the following properties:

1. The current through the input gates is zero.
2. The drain current I and the input-gate voltages V_1 and V_2 are related as $I = I_s \exp(\kappa(V_1+V_2)/U_\mathrm{T})$, where I_s is a pre-exponential scaling constant, κ is a positive dimensionless weight, and U_T is the thermal voltage, kT/q.

3.2 Nonlinear function realization

The implementation of nonlinear functions using translinear circuits is discussed in [13]. Elementary operations like addition and subtraction are easily performed in any current-mode system (using Kirchoff's current law (KCL) and a current mirror, respectively). Translinear circuits, in particular, can also do other elementary operations like multiplication, division, and exponentiation (with rational exponents). Hence any algebraic function can be synthesized by simply expressing it in terms of these elementary operations. Transcendental functions like $\exp(x), \log(x),$ and $\arctan(x)$ are implemented by suitably approximating them using algebraic functions. Different techniques exist for approximating functions by rational functions [14]. Approximation with minimax or near-minimax error is one of the more suitable methods for approximation over an interval. Remez's algorithm [14] is used to determine the minimax rational approximation, while numerous other techniques exist to get near-minimax approximations (Maple's 'minimax' command implements Remez's algorithm). The transcendental functions that need to be implemented in the bearings-only tracking algorithm are the inverse tangent *arctan* and the *Gaussian* $\exp(-x^2/2)$. The approximations and the corresponding implementations of these functions are considered below.

3.2.1 Implementation of the inverse tangent function

The function ϕ to be approximated is as follows (normalized so that $\phi(\infty) = 1$):

$$\phi(x) = \frac{2}{\pi}\arctan(x), \text{ where } |x| < \infty. \tag{7}$$

An approximation of ϕ using algebraic functions, given in [13], is as follows:

$$y = f(x) = \frac{x}{0.63 + \sqrt{0.88 + x^2}}, \text{ where } |x| < \infty. \tag{8}$$

The maximum error obtained using the approximation is less than 0.05% of the maximum value.

The implementation of f in (8) using MITEs is done through the following steps:

1. **Scaling.** Since the input and output variables are represented by currents, to maintain dimensional consistency, the substitutions $x \mapsto I_x/I_a$ and $y \mapsto I_y/I_a$ are done. Hence, we have
$$I_y = \frac{I_x I_a}{0.63 I_a + \sqrt{0.88 I_a^2 + I_x^2}} \tag{9}$$
2. **Current splitting.** Since the input x can take both positive and negative values and since the currents through MITEs must necessarily be positive, we use a current splitter [15] to produce currents I_{x+} and I_{x-} satisfying $I_{x+} - I_{x-} = I_x$ and $I_{x+}I_{x-} = I_a^2$.
3. **Block reduction.** The equation to be implemented thus becomes
$$\begin{aligned} I_y &= \frac{I_{x+}I_a - I_{x-}I_a}{0.63 I_a + \sqrt{-1.12 I_a^2 + I_{x+}^2 + I_{x-}^2}} \\ &= \left(\frac{I_{x+}I_a}{\left(0.63 I_a + \left(\sqrt{I_r I_a}\right)\right)}\right) - \left(\frac{I_{x-}I_a}{\left(0.63 I_a + \left(\sqrt{I_r I_a}\right)\right)}\right)\end{aligned}$$
where $I_r = I_{x+}^2/I_a + I_{x-}^2/I_a - 1.12 I_a$. The parentheses show the order in which the operations are implemented. Each of the blocks (representing the operation in the parentheses) is implemented using procedures described in [12].
4. **Consolidation.** As described in [11], reductant MITEs are removed using *consolidation* to arrive at the final circuit shown in Fig. 2(b).

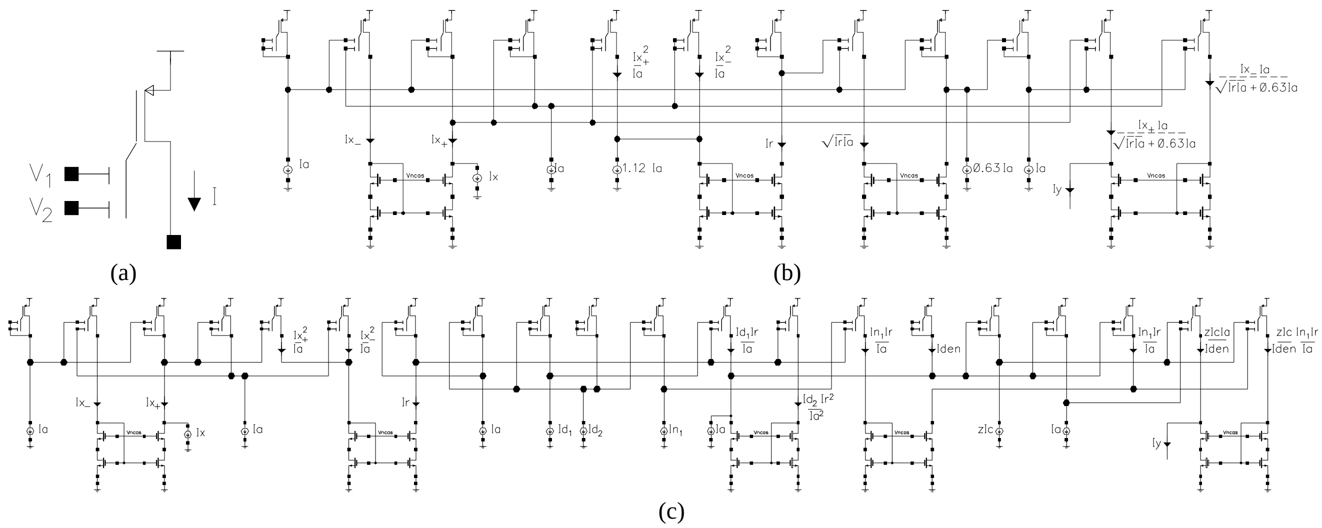

Figure 2: (a) Symbol for a 2-input MITE. Ideally, it should obey the law $I = I_s \exp(\kappa(V_1 + V_2)/U_T)$. (b) The MITE circuit used to implement the arctan function. The current $I_r = I_{x+}^2/I_a + I_{x-}^2/I_a - 1.12 I_a$ and the output is given by $I_y = (I_{x+}I_a - I_{x-}I_a)/(\sqrt{I_a I_r} + 0.63 I_a)$. (c) The MITE circuit used to implement the Gaussian function. The intermediate variables used are given by $I_r = I_{x+}^2/I_a + I_{x-}^2/I_a$, and $I_{den} = I_a - (I_{d1}I_r)/I_a + (I_{d2}I_r^2)/I_a^2$. The output is $I_y = zI_c(I_a - (I_{n1}I_r)/I_a)/I_{den}$.

3.2.2 *Implementation of the Gaussian*

After scaling and normalization, the Gaussian is transformed into $I_y = I_c \exp(-I_x^2/(2I_a^2))$. A current splitter converts I_x into two positive currents I_{x+} and I_{x-} satisfying $I_{x+} - I_{x-} = I_x$ and $I_{x+}I_{x-} = I_a^2$. Hence, we have $I_x^2 = I_{x+}^2 + I_{x-}^2 - 2I_a^2 = I_r I_a - 2I_a^2$, where $I_r = I_{x+}^2/I_a + I_{x-}^2/I_a$. Thus, $I_y = eI_c \exp(-I_r/(2I_a))$. It should also be noted that if the implementation is to be valid for $I_x \in [-bI_a, bI_a]$, then it suffices to approximate $\exp(-I_r/(2I_a))$ for $I_r \in [2I_a, (2 + b^2)I_a]$. The minimax rational approximation for $b = 4$ with the numerator and denominator degrees equal to 1 and 2, respectively, was found using Remez's algorithm and is given by :

$$I_y = zI_c \frac{I_a - I_{n1}(I_r/I_a)}{I_a - I_{d1}(I_r/I_a) + I_{d2}(I_r^2/I_a^2)} \qquad (10)$$

where $I_r = (I_{x+}^2 + I_{x-}^2)/I_a$, $I_{n1} = 0.07195 I_a$, $I_{d1} = 0.2913 I_a$, $I_{d2} = 0.1641 I_a$, and $z = 1.245$. The computations that involve multiplication are done using a new synthesis procedure especially meant for such "product–of–power–law" computations [16]. The currents I_{n1}, I_{d1}, I_{d2} are set using programmable floating–gate MOSFETs [17]. The final circuit is shown in Fig. 2(c).

4. PARTICLE FILTER IMPLEMENTATION

4.1 Architecture

Particle filter algorithms use nonlinear functions and operate on scalar or vector data, as opposed to blocks of data as in speech or image processing. Their complexity is related to the number of particles N and the nonlinear functions used. Most applications that use particle filters perform a Gaussian evaluation at the weighting stage. Depending on the state-space model used, some applications might use additional nonlinear functions in the particle proposal and weight evaluation stage. Except for the state estimate and resampling stage, processing of individual particles can be done in parallel. We concentrate on the computations performed on a single particle at the weight evaluation stage.

In this work, we propose to use analog computations in the weight evaluation stage of a bearings-only tracker. As shown in Fig. 3, this involves simple operations such as addition, multiplication, and nonlinear operations such as *Gaussian* and *arctan* evaluation. The use of analog computation leads to a mixed-mode implementation of the particle filter algorithm, where computations

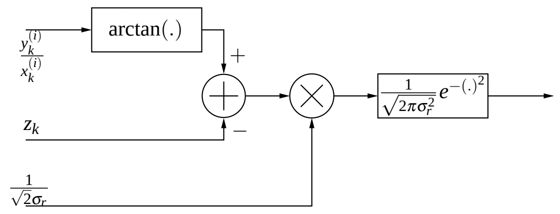
arctan(.)
$y_k^{(i)} / x_k^{(i)}$
+
−
z_k
$\frac{1}{\sqrt{2}\sigma_r}$
$\frac{1}{\sqrt{2\pi\sigma_r^2}} e^{-(.)^2}$

Figure 3: Computations at the weight evaluation stage of the particle filter algorithm used in bearings-only tracking.

are performed in both the analog and digital domains. The mixed-mode implementation leads to two possible methods that differ in the way the analog-digital partition is formed.

In Method-1, shown in Fig. 1, the weight evaluation stage is implemented in the analog domain and the remaining stages in the digital domain. This uses data converter blocks, D-to-A and A-to-D, before and after the weight update stage. The minimum number of bits to be used in the D-to-A and A-to-D is based on physical requirements. The lowest value to be represented must be above the noise level in the analog circuit. Increasing the number of bits in these blocks can increase latency and power consumption. Hence, a compromise among accuracy, speed, and power consumption has to be made such that the power savings from the analog computations is not offset. However, a detailed treatment of these trade-offs is beyond the scope of this paper.

In Method-2, shown in Fig. 4, the resampling stage is implemented in the digital domain and the remaining stages in the analog domain. This method uses fewer D-to-As and A-to-Ds compared to Method-1. The measurements, angles in a bearings-only tracker, may be from a source localization algorithm implemented in the analog domain [18]. The resampling stage assumes the availability of analog memory, whose access can be controlled using a digital controller. Other operations in the proposal and state estimation stages can provide significant power savings compared to Method-1. The proposal stage uses four additions to implement (2). In the analog domain using MITE networks, the addition of two signals is based on KCL and performed by connecting the wires carrying the corresponding currents. In the digital domain, addition is performed using adder circuits. The state estimation, Step 4 in Table 1, can be achieved using a vector multiplier in the analog domain.

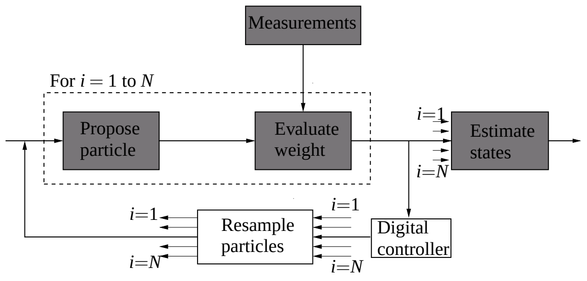

Figure 4: Block diagram showing computational flow in the proposed implementation Method-2. Highlighted stages are performed in the analog domain.

Our simulations use Method-1 in a bearings-only tracker. An issue with Method-2 is generating random samples in the proposal stage. If this stage has to be implemented in analog, we need a robust way to achieve this. Another issue is the use of analog memory in the resampling stage. Method-2, however, can provide significant power savings compared to Method-1. In either method, the reference currents are chosen such that the MITE networks remain in their valid region of operation.

4.2 Power consumption

In this section, the power consumed by the proposed analog implementation and a digital implementation are presented. From (5) and (6) we see that significant computations in the particle weighting stage of the particle filter algorithm involve the evaluation of *arctan* and *Gaussian* functions. These functions also limit the speed of the particle filter algorithm in a digital implementation [4], [5]. Hence, the power consumed in these computations is considered here.

4.2.1 Digital implementation

Power-aware design in digital circuits is an active area of research [19]. Power consumption in CMOS circuits consists of dynamic and static power. Ignoring the negligible static power, the average dynamic power consumed can be approximated, [20], by $P = \alpha_{0\to1} C_L V_{dd}^2 f_{\text{clk}}$, where $\alpha_{0\to1}$ is the node transition activity factor, C_L is the load capacitance, V_{dd} is the supply voltage and f_{clk} is the clock frequency.

Digital implementations of the *arctan* or *Gaussian* function use the CORDIC algorithm. The CORDIC algorithm is an iterative algorithm that can be used to implement trigonometric and hyperbolic functions. The *Gaussian* function can be implemented using the relation $e^x = \sinh(x) + \cosh(x)$. There are several implementation architectures of the CORDIC block depending on performance requirements. The basic computation, in a single iteration or pipeline, involves two shifts, three additions, and one look-up operation. Its complexity and speed depends on the required m bits of precision.

A VHDL implementation of a pipelined CORDIC block was simulated, and estimated power consumption using the AMI 0.35μm CMOS process is shown in Table 2. For reference, results from a low-power CORDIC implementation using MOS Current Mode logic (MCML) demonstrated in [22] is also shown. Their implementation specifications, however, are different.

4.2.2 Analog implementation

The analog implementation of the *arctan* and *Gaussian* functions use MITEs as explained in Sec. 3. The instantaneous power consumed by these circuits depends upon the (instantaneous) value of the input current. For the *arctan* circuit, the current drawn from the supply can be shown to be

$$I_a\left(9.26 + \frac{2I_x^2}{I_a^2} + \sqrt{4+\frac{I_x^2}{I_a^2}} + 2\sqrt{\frac{I_r}{I_a}} + \frac{\sqrt{4+I_x^2/I_a^2}}{\sqrt{I_r/I_a}+.63}\right)$$

Table 2: Characteristics of digital implementation

Implementation	Digital	MCML [22]
$\mathbf{V_{DD}}$(V)	3.3	1.0
Clock Frequency (MHz)	1	125
Output precision (bits)	12	8
Power (mW) [a]	0.55 x 2	4.33 x 2

[a]The factor of 2 is to account for two CORDIC blocks.

Table 3: Characteristics of analog implementation

Circuit	*arctan*		*Gaussian*	
	Minimum	Maximum	Minimum	Maximum
Ref. I_a (nA)	0.5	20	0.1	10
Input current	$-10I_a$	$10I_a$	$-10I_a$	$10I_a$
Power (μW)	0.361	14.45	1.097	109.7
Error (%)	0.31	0.71	2.04	2.8

Table 4: Simulation parameters

Tracker parameters			**Circuit parameters** (variables in (6))		
N	σ_r	σ_u	$I_{\text{in}}(y_k^{(i)}/x_k^{(i)})$	Ia_{arctan}	$Ia_{\text{Gaussian}}(\sigma_r)$
1000	0.29°	0.001	6.36nA to 7.18nA	10nA	0.1nA

Assuming $|I_x|_{\max} = 10I_a$, the maximum power dissipated is $240.82 I_a V_{dd}$. For a typical value of $I_a = 10$nA with a 3V supply, the maximum power consumed is 7.23μW. A similar expression can be derived for the *Gaussian* block. Assuming that $I_c = I_a$, the maximum power consumed is $3655.14 I_a V_{dd}$. For $I_a = .1$nA the power consumed to is 1.097μW.

An accurate comparison, based on power consumption, between the analog and digital implementation can only be made after performing transient and signal–to–noise ratio (SNR) analyses of the analog implementation. These analyses are part of ongoing research.

5. SIMULATION

The circuit blocks were simulated for different values of the input currents and the reference currents to determine the accuracy of the implementation. Models for the AMI 0.5μm CMOS process were used in the simulations. The results are shown in Table 3. The error (as a percentage of the maximum) and power values are over the ranges of the input and reference currents shown. The analog circuit for the weighting stage of the bearings-only tracker was simulated and results compared to a Matlab simulation. The range for input currents was obtained from the Matlab simulation. The simulation parameters used are similar to those in [9] and are shown in Table 4. The plot in Fig. 5 shows the weights obtained from the analog circuit simulation and the approximation used. The error introduced by the *arctan* block shifts the mean of the *Gaussian* distribution in the weight evaluation. From Table 3, the error in the *arctan* block corresponds to an error of 0.28° to 0.64°. Depending on the measurement noise variance σ_r, the shift in the peak may or may not affect the state estimate. Further, the implemented *arctan* block has quadrant ambiguity and hence additional logic needs to be included to have results in the correct quadrant. The error in the regions near $\pm 90°$ is relatively high compared to those near $0°$.

6. DISCUSSION AND FUTURE WORK

The different currents in the circuit and the sizes of the MITEs must be chosen so that all the MITEs remain in the sub-threshold region.

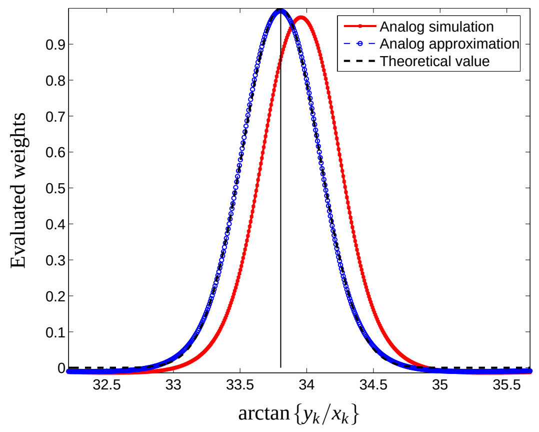

Figure 5: Comparison of the weights evaluated using (a) the analog circuit, (b) the functional approximation, and (c) the theoretical value. The shift in the peak is due to an error of 0.15° introduced by the arctan block. (The parameter $\sigma_r^2 = \{0.29^\circ\}^2$ and $z_k = 33.8^\circ$ in (6)).

Each of the reference currents in the blocks are tunable, thanks to the programming capability of floating–gate MOSFETs. Thus, the reference currents can be set as low as desired as long as other specifications such as bandwidth and SNR are met. As MITEs are typically implemented using floating–gate MOSFETs, care should be taken to ensure that the floating–gate capacitors do not render the circuits unstable. Some input waveforms were seen to produce undesirable oscillations. This requires frequency compensation in the analog blocks to ensure unconditional stability for all (relevant) input waveforms. Research to alleviate such problems is in progress. The bandwidth of operation expected from these blocks is determined by the rate at which the output estimates are required. This rate also determines the clock frequency of the digital controller in Method-2 shown in Fig. 4. While the implementation presented is low-power, a direct comparison with the digital implementation requires both of them to run at the same clock frequency with comparable accuracy. Estimating the delay in the analog blocks is difficult since it depends upon the input values (which cause oscillations in certain cases). Hence, this is indeed a viable solution but requires further work.

It should be noted that if the processing were done in continuous time as opposed to discrete time, all the blocks can be analog. However, this necessitates a complete change in the way we view Bayesian estimation problems.

7. CONCLUSIONS

In this paper, we proposed a low-power, analog and mixed-mode implementation for particle filters. An implementation method that uses minimal number of data converters was presented and is more promising. This was partly demonstrated by implementing the nonlinear functions, *Gaussian* and *arctan*, in the analog domain using MITEs. These functions were used in a bearings-only target tracker that uses particle filters. While this is a low-power implementation, some operational issues need to be addressed before characterizing the overall behaviour of the analog blocks. As part of future research, we will analyze the performance and power consumption of the system as a whole including the data conversion blocks.

REFERENCES

[1] A. Doucet, N. Freitas, and N. Gordon, eds., *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[2] P. Hasler, “Low-power programmable signal processing,” in *Proc. 5th IEEE Intl. Workshop on System-on-Chip for Real-Time Appl.*, (Banff, Alberta - Canada), pp. 413–418, July 2005.

[3] B. A. Minch, C. Diorio, P. Hasler, and C. A. Mead, “Translinear circuits using subthreshold floating-gate MOS transistors,” *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 2, pp. 167–179, 1996.

[4] M. Bolic, *Architectures for efficient implementation of particle filters*. PhD thesis, Stony Brook University, New York, August 2004.

[5] S.-S. Chin and S. Hong, “VLSI design of high-throughput processing element for real-time particle filtering,” *Intl. Symp. Signals, Circuits and Systems*, vol. 2, pp. 617–620, July 2003.

[6] S. Hong, X. Liang, and P. M. Djuric, “Reconfigurable particle filter design using dataflow structure translation,” *2004 IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 325 – 30, 2004.

[7] M. Shabany, H. Shojania, J. Zhang, J. Omidi, and P. G. Gulak, “VLSI architecture of a wireless channel estimator using sequential monte carlo methods,” *2005 IEEE 6th Workshop on Signal Processing Advances in Wireless Communications*, pp. 450–454, June 2005.

[8] O. Brun, V. Teuliere, and J.-M. Garcia, “Parallel particle filtering,” *J. Parallel and Distributed Comput.*, vol. 62, pp. 1186–1202, July 2002.

[9] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, “Novel approaches to nonlinear/non-Gaussian Bayesian state estimation,” *IEE Proc.-F*, vol. 140, pp. 107–113, April 1993.

[10] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. Academic-Press, 1988.

[11] B. A. Minch, “Synthesis of static and dynamic multiple-input translinear element networks,” *IEEE Trans. Circuits Syst. I*, vol. 51, pp. 409–421, Feb. 2004.

[12] B. A. Minch, “Construction and transformation of multiple-input translinear element networks,” *IEEE Trans. Circuits Syst. I*, vol. 50, pp. 1530–1537, Dec. 2003.

[13] E. Seevinck, *Analysis and Synthesis of Translinear Integrated Circuits*. Amsterdam: Elsevier, 1988.

[14] C. T. Fike, *Computer Evaluation of Mathematical Functions*. Series in Automatic Computation, Prentice Hall, Inc., 1968.

[15] D. Frey, “Current mode class AB second order filter,” *Electronics Letters*, vol. 30, pp. 205 – 206, Feb. 1994.

[16] S. Subramanian, D. V. Anderson, P. Hasler, and B. A. Minch, “Optimal synthesis of MITE translinear loops.” Technical Report CADSP-TR-2006-1, available at `http://cadsp.ece.gatech.edu/Publications/tr_1.pdf`, 2006.

[17] G. Serrano, P. D. Smith, H. J. Lo, R. Chawla, T. S. Hall, C. Twigg, and P. Hasler, “Automatic rapid programming of large arrays of floating-gate elements,” in *Proc. Intl. Symp. on Circuits and Syst.*, vol. 1, pp. I–373 – I–376, May 2004.

[18] M. Stanacevic and G. Cauwenberghs, “Micropower gradient flow acoustic localizer,” in *IEEE Trans. Circuits Syst. I*, vol. 52, pp. 2148–2157, Oct 2005.

[19] M. Pedram and J. M. Rabaey, eds., *Power aware design methodologies*. Kluwer Academic Publishers, 2002.

[20] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

[21] S. Wang, V. Piuri, and E. E. S. Jr., “A unified view of CORDIC processor design,” in *Proc. IEEE 39th Midwest Symp. Circuits and Syst.*, vol. 2, pp. 852–855, 1996.

[22] J. Musicer and J. M. Rabaey, “MOS current mode logic for low power, low noise CORDIC computation in mixed-signal environments,” in *Proc. Intl. Symp. Low Power Electronics and Design*, pp. 102–107, 2000.