IMPLEMENTATION OF BATCH-BASED PARTICLE FILTERS FOR MULTI-SENSOR TRACKING

Rajbabu Velmurugan[*], *Volkan Cevher*[†], *and James H. McClellan*[*]

[*]Georgia Institute of Technology, Atlanta, GA 30332
[†]University of Maryland, College Park, MD 20742

ABSTRACT

In this paper, we demonstrate fixed-point FPGA implementations of state space systems using Particle Filters, especially multi-target bearing and range tracking systems. These trackers operate either as independent organic trackers or as a joint tracker to estimate a moving target's state in the x-y plane. For the efficiency of the particle filter, we consider factorized posterior approximations based on the Laplacian approximation, which uses a Newton-Raphson search. We delineate the computation and memory resources needed for real-time performance of the range and bearing particle filter trackers. Our implementations are demonstrated using the Xilinx System Generator. As part of the FPGA implementation, a floating-point, soft- and hard-core implementation of the Newton search algorithm is also developed.

Index Terms— Particle filter, implementation, multi-sensor tracking

1. INTRODUCTION

Multi-target tracking is an important problem in sensor networks [1]. In the scenarios considered here, acoustic direction-of-arrival (DOA) measurements or range measurements received at a single, stationary sensor are used to track multiple, maneuvering, locally constant velocity targets. The nonlinear state and observation models, and non-Gaussian noise in the system are handled by using a particle filter approach to estimate the target states. The data association problem in multi-target tracking is addressed by means of an image-template matching approach that uses batch measurements. This approach was introduced in [2] and also used in [3], [4]. Another scenario considered in this paper is the case where the range and acoustic sensors are collocated leading to a joint tracking system to track targets [5]. The focus of this paper is the FPGA implementation of these particle filter tracking algorithms. The proposal function used in the trackers considered is an approximation to the full-posterior and obtained by applying Laplace's method. Laplace's method in turn, uses a Newton-Raphson search to solve a convex optimization problem and evaluate the mode and variance of a distribution. The Newton search in the proposal stage is implemented as a soft-core or hard-core in the FPGA device. This approach is different from earlier FPGA implementation of particle filters that considered a bootstrap particle filter to perform bearings-only tracking [6, 7]. The proposal function in a bootstrap filter is the state update and hence the implementation requirements were different and less complex. Further, in our approach we use the Xilinx System Generator [8] to develop the FPGA implementation.

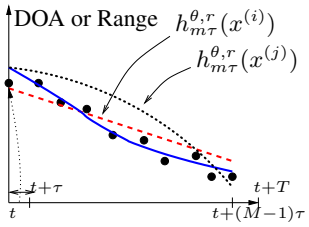

Fig. 1: Template matching idea is illustrated. The solid line represents the true DOA track. Black dots represent the noisy DOA estimates. The dashed and dotted lines represent the DOA tracks for two proposed particles, $x^{(i)}$ and $x^{(j)}$. These tracks are calculated using the state update function h. Visually, the i-th particle is a better match than the j-th particle; hence, its likelihood is higher.

2. BATCH MEASUREMENT-BASED TRACKING

The particle filter developed here uses batch measurements to estimate the state vector, based on an image template-matching idea. We denote the collection of M measurements a batch, where M is the batch size. In our DOA-based tracking problem, a temporal DOA image is first formed when a batch of DOA observations is received from a beamformer that processes the received acoustic data at M τ-second intervals. Similarly in range-based tracking a temporal range image is formed using a radar sensor. Then, image templates for target tracks are created using the state update function and the target state vectors (Fig. 1). By determining the best matching template (e.g., most probable target track), the target state vectors are estimated. Because the observations are treated as an image, the data association and measurement ordering problems are naturally alleviated. Moreover, by assuming that the observations are approximately normally distributed around the true target tracks, with constant miss-probability and clutter density, a robust particle filter tracker is formulated.

3. MULTI-TARGET TRACKING ALGORITHMS

Here, we provide a brief description of the particle filter target tracking algorithms.

3.1. DOA-only tracking

In the DOA-only tracking algorithm [3], the measurements are a batch of DOAs obtained from an acoustic sensor array. The particle filter state vector $\mathbf{x}_t = \left[\, x_1^T(t),\ x_2^T(t),\ \cdots,x_K^T(t) \,\right]^T$ consists of the concatenation of partitions $x_k(t)$ for each target, indexed by k

and K is the number of targets at time t. Each partition has the corresponding target motion parameters $x_k(t) \triangleq [\,\theta_k(t)\,,\ Q_k(t)\,,\ \phi_k(t)\,]^T$, where $\theta_k(t)$ is DOA, $\phi_k(t)$ the heading direction, and $Q_k(t) = \log(v_k/r_k(t))$ the logarithm of velocity over range. Assuming a locally constant velocity model for target motion [2], the resulting state update equation is:

$$x_k(t+T) = h_{\theta,T}\left(x_k(t)\right) + u_k(t), \tag{1}$$

where $u_k(t) \sim \mathcal{N}(0, \Sigma_u)$ with $\Sigma_u = \mathrm{diag}\{\sigma^2_{\theta,k}, \sigma^2_{Q,k}, \sigma^2_{\phi,k}\}$ and $h_{\theta,T}(x_k(t))$ is

$$\begin{bmatrix} \tan^{-1}\left\{\frac{\sin\theta_k(t)+T\exp Q_k(t)\sin\phi_k(t)}{\cos\theta_k(t)+T\exp Q_k(t)\cos\phi_k(t)}\right\} \\ Q_k(t) - \frac{1}{2}\log\big\{1 + 2T\exp Q_k(t)\cos(\theta_k(t)-\phi_k(t)) \\ \qquad + T^2\exp(2Q_k(t))\big\} \\ \phi_k(t) \end{bmatrix}. \tag{2}$$

The observation $\mathbf{y}_t$ at time t, consists of the batch DOA estimates from a beamformer, given by

$$\mathbf{y}_t = \{y_{t+m\tau}(p)\}_{m=0}^{M-1}, \quad p = 1, \ldots, P \tag{3}$$

where m is the batch index and P the number of peaks. With this observation model, a data likelihood function for the multi-target tracking scenario considered was introduced in [2]. This approach is similar to the model used in visual tracking [9]. It assumes that the batch of DOAs form a normally distributed cloud around the true target DOA tracks with a constant miss probability and may have spurious peaks that are Poisson distributed. Of the P peaks, only one peak corresponds to a target and the remaining peaks either correspond to other targets or clutter. Hence, for a specific time instant within a batch period, say $m = m_i$, we write the observation density as: $p(\mathbf{y}_m(t)|\mathbf{x}_k(t)) \propto$

$$1 + \frac{1-\kappa}{\sqrt{2\pi\sigma^2_\theta(m_i)}\kappa\lambda}\sum_{p_i=1}^{P}\exp\left\{-\frac{\left(h^\theta_{\theta,m_i\tau}\left(x_k(t)\right) - y_{t+m_i\tau}(p_i)\right)^2}{2\sigma^2_\theta(m_i)}\right\}, \tag{4}$$

where κ is a constant miss probability, $\lambda = \dfrac{\gamma}{2\pi}$ is the Poisson rate of the clutter distribution, superscript θ on the state update function $h_{\theta,t}$ refers to the DOA component of the state update (2), and $\sigma^2_\theta(m)$ is supplied by a beamformer. Extending this to a batch of M observations for a single target, we have

$$p(\mathbf{y}(t)|\mathbf{x}_k(t)) \propto \prod_m p(\mathbf{y}_m(t)|\mathbf{x}_k(t)). \tag{5}$$

Hence, the observation density for multiple targets is: $p(\mathbf{y}(t)|\mathbf{x}(t)) \propto \prod_k \prod_m p(\mathbf{y}_m(t)|\mathbf{x}_k(t))$.

The state update and data likelihood functions derived here can be used in a particle filter algorithm to estimate the multi-target motion parameters. If the initial target x-y positions are known, the motion parameter estimates can then be used to estimate the target positions. The particle filter algorithm is discussed in Section 4.

3.2. Range-only tracking

Here the batch measurements are target ranges obtained using a Radio Frequency (RF) sensor [4]. The particle filter state vector $\mathbf{x}_t$ consists of the concatenation of the individual target motion vectors $x_k(t)$ defined as $[\theta_k(t),\ R_k(t),\ v_k(t),\ \phi_k(t)]^T$, where the target DOA is $\theta_k(t)$, range (logarithm of) from sensor is $R_k(t)$, velocity is $v_k(t)$, and heading direction is $\phi_k(t)$. Similar to (1), the state update in the range-only tracker is:

$$x_k(t+T) = h_{R,T}\left(x_k(t)\right) + u_k(t), \tag{6}$$

where $u_k(t) \sim \mathcal{N}(0, \Sigma_u)$ with $\Sigma_u = \mathrm{diag}\{\sigma^2_{\theta,k}, \sigma^2_{r,k}, \sigma^2_{v,k}, \sigma^2_{\phi,k}\}$ and $h_{R,T}(x_k)$ is

$$\begin{bmatrix} \tan^{-1}\left\{\frac{e^{R_k}\sin\theta_k + Tv_k\sin\phi_k}{e^{R_k}\cos\theta_k + Tv_k\cos\phi_k}\right\} \\ \frac{1}{2}\log\left\{e^{2R_k} + T^2v_k^2 + 2Te^{R_k}v_k\cos(\theta_k - \phi_k)\right\} \\ v_k \\ \phi_k \end{bmatrix}. \tag{7}$$

The observations are range measurements that have the same batch structure as the acoustic DOA measurements. Hence, the data likelihood for the range tracker is similar to (5) with $h^\theta_{\theta,m\tau}\left(x_k(t)\right)$ replaced by $h^r_{R,m\tau}\left(x_k(t)\right)$, y_t denotes range measurements, and the noise parameters depend on the range tracker. Adopting an approach similar to DOA-only tracking, the range-only tracker can also be used to track x-y positions of multiple targets.

3.3. Joint radar-acoustic tracking

Using co-located range and acoustic sensors will directly provide target position in Cartesian coordinates. Such a joint tracker has been developed in [5]. In this tracker the measurements are batch of DOAs and range measurements. The particle filter target state vector $z_t = [x_t, y_t, v_{x,t}, v_{y,t}]^T$ consists of the target position and velocity in Cartesian coordinates. For locally linear target motion the state update is a linear relation given by

$$\mathbf{z}_t = \mathbf{A}_T\mathbf{z}_{t-T} + \mathbf{u}_t, \tag{8}$$

where $\mathbf{u}_t$ is $\mathcal{N}(0, \Sigma_z)$ and $\mathbf{A}_T$ is the constant velocity transition matrix.

Given the target state, the batch of radar and acoustic observations are independent and represented as $\mathbf{y}_t = [\mathbf{y}_{\theta,t}, \mathbf{y}_{R,t}]$. Hence, the data likelihood is a product of the individual likelihoods of the acoustic and radar modalities:

$$p(\mathbf{y}_t|\mathbf{z}_t) = p(\mathbf{y}_{\theta,t}|\mathbf{z}_t)p(\mathbf{y}_{R,t}|\mathbf{z}_t). \tag{9}$$

Assuming batch measurements, missing data, and clutter as described in the acoustic tracker, the individual data likelihoods in (9) are similar to the data likelihood in (5). The difference between them is in the $h^\theta_{\theta,m\tau}\left(x_k(t)\right)$ and $h^r_{R,m\tau}\left(x_k(t)\right)$ functions, because of the difference in target state vector. Because of the huge difference in velocities of electromagnetic and acoustic waves, the range and DOA measurements in the co-located sensor at a specific time instant will correspond to the target's state at different times, so the data likelihood [5] has to be adjusted. Based on this time-delay, the range and DOA update functions in the joint tracker are:

$$h_{R,m}(\mathbf{z}_t) = \sqrt{(x_t + (m-1)\tau v_{x,t})^2 + (y_t + (m-1)\tau v_{y,t})^2},$$
$$h_{\theta,m}(\mathbf{z}_t) = \tan^{-1}\left\{\frac{y_t + (m-1)\tau v_{y,t}}{x_t + (m-1)\tau v_{x,t}}\right\}.$$

Another difference between the joint and the organic particle filter trackers is in the weighting stage. In addition to the usual weighting procedure at the current time instant, there is a pre-weighting stage that reevaluates the posterior from the previous time. This pre-weighting uses a subset of the acoustic data that has information about the previous target state, and involves a data likelihood evaluation similar to that in (5). The details on handling the delay compensation, pre-weighting, and weighting are in [5].

4. PARTICLE FILTER ALGORITHM

Most stages of the particle filter algorithm used in the DOA-only, range-only, and joint tracking algorithms follow the conventional sequential importance resampling (SIR) particle filter [10] and are presented in [3, 4], and [5]. However, the proposal function that determines the efficiency of the algorithm differs from the conventional approach. Hence we only present the proposal function. In all the tracking algorithms considered, the proposal function is an approximation to the full-posterior density and is denoted as:

$$g(x_t|\mathbf{y}_t, x_{t-T}) \approx p(x_t|\mathbf{y}_t, x_{t-T}) \propto p(\mathbf{y}_t|x_t)p(x_t|x_{t-T}), \quad (10)$$

where $p(x_t|x_{t-T})$ can be obtained from the state update in (2), (7), or (8). Assuming Gaussian state noise, obtaining the state distribution does not pose any difficulty. However, the data likelihood $p(\mathbf{y}_t|x_t)$, due to the batch measurements, is nonlinear and hence needs to be approximated. Laplace's method can be used to approximate each factor in the factorized approximation $p(\mathbf{y}_t|x_t)$ in (5) and thereby derive the partition proposal functions of the particle filter, denoted as $g(x_t|\mathbf{y}_t, x_{t-T})$. Laplace's method provides an analytical approximation for pdfs, based on a Gaussian approximation of the density around its mode, where the inverse Hessian of the logarithm of the density is used as a covariance approximation [11].

The Laplacian approximation is described in [2] and is implemented using the Newton-Raphson recursion with backtracking for computational efficiency. The final expression for the partition proposal functions to be used in the particle filter is given by

$$g_k(x_k(t)|\mathbf{y}_t, x_k(t-T)) \sim \mathcal{N}\left(\mu_g(k), \Sigma_g(k)\right) \quad (11)$$

where the Gaussian density parameters are

$$\begin{aligned} \Sigma_g(k) &= \left(\Sigma_y^{-1}(k) + \Sigma_u^{-1}\right)^{-1} \\ \mu_g(k) &= \Sigma_g(k)\left(\Sigma_y^{-1}(k)x_{k,\text{mode}} + \Sigma_u^{-1}h_T(x_k(t-T))\right), \end{aligned} \quad (12)$$

where $x_{k,\text{mode}}$ is the mode of $p(\mathbf{y}_t|x_k(t))$, and $\Sigma_y^{-1}(k)$ is the Hessian of $p(\mathbf{y}_t|x_k(t))$ at $x_{k,\text{mode}}$, obtained using the Newton search.

5. FPGA IMPLEMENTATION OF CERTAIN SECTIONS

In this section, we develop an FPGA implementation for the various stages in the particle filter algorithm using the Xilinx System Generator [8]. We also present an FPGA based soft- and hard-core strategy to implement the Newton-Raphson search in the particle proposal stage. The particle resampling is performed using the systematic resampling described in [6]. The individual stages of the particle filter algorithm in the DOA tracker are implemented and verified by comparing with Matlab simulation results. A full implementation will involve additional high level control circuits and memory blocks for data flow among the stages and are not considered here. However, implementing individual stages illustrates the key differences between the batch-based particle filter that uses a near optimal proposal function to that of a bootstrap particle filter tracker that uses the state transition to propose particles [6], [7].

5.1. Fixed-point simulation

Most stages of the particle filter algorithm, except the Newton search, were implemented and verified using fixed-point data. The choice of target states and measurement variances affect the required word length. A detailed description on the choice of fixed-point word lengths for the DOA-only tracker is provided in [12]. From the Matlab fixed-point simulations of the DOA-only tracker, the fixed-point word length used is 16 bits with 10 bits for the fractional part. For the range-only tracker, the range measurements and their exponential values can be high and this increases the required word length to 25 bits. Otherwise its implementation closely follows the DOA-only tracker. As for the joint tracker, since the target states are in Cartesian co-ordinates, a word length of 16 bits is sufficient.

5.2. Particle state update

The implementation of the organic particle state update function in (2) and (7) involves updating the states θ, Q, and R because the other states are propagated directly. The trigonometric, exponential, and logarithmic functions were implemented using the CORDIC algorithm. The CORDIC implementation in the Xilinx System Generator uses a fully parallel and pipelined architecture. On the other hand, the state update function (8) in the joint tracker does not involve any nonlinear functions leading to reduced FPGA resources.

5.3. Data likelihood evaluation

The data likelihood function (5) will be used in both the particle proposal stage and the weight evaluation stage. Apart from the number of nonlinear functions to be evaluated, the use of batch-measurements is a significant difference between this tracker and the bootstrap bearings-only tracker [6]. For a single-target, there are two ways in which the batch computations in (5) can be performed. They can be done in parallel using M computation units or using a single unit with sequential updating of the product term. The parallel implementation of the data likelihood consumes nearly M times more resources when compared to the sequential implementation. Hence, a sequential approach is adopted in this research. In the joint tracker, the data likelihood (9) is a product of the radar and acoustic data likelihoods leading to increased resources in the likelihood evaluation.

5.4. Particle weight evaluation

The evaluation of the i-th particle weight using

$$w^{*(i)}_t = w^{(i)}_{t-T}\frac{p(\mathbf{y}_t|\mathbf{x}^{(i)}_t)p(\mathbf{x}^{(i)}_t|\mathbf{x}_{t-T})}{\prod_k g_k(x^{(i)}_k(t)|\mathbf{y}_t, x^{(i)}_k(t-T))}, \quad (13)$$

involves evaluating the state transition distribution from (2), (6), or (8), the proposal distribution (11), the data likelihood (5), and an exponential. The operations involved in evaluating the Gaussian state and proposal distribution probabilities are similar and hence the corresponding blocks can be reused.

5.5. Newton-Raphson search

In Section 4, Laplace's method is used to approximate the data likelihood as a Gaussian. This is done by executing a Newton-Raphson search to identify the mode and curvature of the distribution. The Newton-Raphson recursion is given by the familiar expression $x^{l+1}_k = x^l_k - \nu_l H^{-1}_l G_l$, where ν is the algorithm step size, $G = \partial J/\partial x_k$ the gradient, and $H = \partial^2 J/\partial x_k \partial x^T_k$ the Hessian. For the DOA-only tracker, the cost function J to be minimized is similar to (6) in [2]. This cost function is an approximation to the full posterior that mitigates numerical sensitivity in the Newton search. See [4] and [5] for similar cost functions for the the range-only and joint trackers.

Table 1: FPGA resource utilization for the batch-based trackers.

Tracker	# Slices[†b] in PF Stage			
	Proposal[†a]	Weight evaluation	Resampling	Overall
DOA-only	7803	8869	374	17046
Range-only	16784	13434	374	30592
Joint	6468	11866	374	18708

[†a] Excluding the Newton search and Gaussian noise generation.
[†b] For comparison, a 16 bit multiplier uses 153 slices

Table 2: Latency at various stages of the trackers.

Tracker	Latency in clock cycles			
	Proposal[†a]	Weight evaluation	Resampling	Overall
DOA-only	168	134	$N + 5$	$3N + 307$
Range-only	214	180	$N + 5$	$3N + 399$
Joint	156	181	$N + 5$	$3N + 342$

[†a] Excluding the Newton search.

A generic Newton-Raphson search itself is not difficult to implement in an FPGA. Mathematical operations such as division and square-root are themselves implemented in hardware using Newton-Raphson search [13]. However, the cost function here contains nonlinear function evaluations that significantly increase the complexity. If the iteration is implemented in fixed-point, special care is needed to address precision, and guarantee convergence. Instead, we implement the Newton search in software using floating-point on a soft-core or hard-core processor that is a part of an FPGA.

The Xilinx Virtex II Pro device [14] used here offers a Xilinx MicroBlaze soft-core and a PowerPC based hard-core. When using a soft- or a hard-core to implement the Newton-Raphson search is that the C code developed for a floating-point DSP implementation can be reused with only minor modifications. We achieve this using the Xilinx Platform Studio (XPS) and embedded development kit (EDK) [15]. One advantage of using a soft-core implementation, over a hard-core, is that the components of the designed CPU can be user specified. Further, the MicroBlaze can be configured to have a faster floating point unit (FPU). However, the current version of the MicroBlaze supported on the Xilinx XCV2P30 board has a local memory of 64 KB available to store the program which is smaller than the 110 KB required by the Newton-Raphson algorithm. In the future, we will configure a recent version of the processor to use additional scratchpad memories [16] for the Newton search. At present, we have used the hard-core processor to achieve software implementation of the Newton search on the FPGA. While we performed a functional verification of the Newton-Rapshon search, specific performance or time profiling will be part of future research.

5.6. Summary of resource usage and latency

Here, we summarize the estimated resource utilization and latency for the organic batch-based particle filter algorithm, because the entire batch-based tracker was not actually implemented in the FPGA. These resource estimates are for a single particle evaluation in the single-target case, and were obtained using the *Resource Utilization* block in the Xilinx System Generator. The resources and latency for the Newton search algorithm, implemented in hard-core, are not included in this summary. When compared to the resources utilized by a bootstrap bearings-only tracker [12], the batch-based tracker uses nearly four times more resources. The total number of required slices in Table 1 is more than the available resource (13, 696 slices) in the FPGA device [14] originally considered in this research. However, the Xilinx Virtex-5 LX330 devices have 51, 840 slices and would be able to accommodate the individual algorithms.

The overall latency (Table 2) in the batch-based tracker depends on the number of particles N. The additional latency of $2N$ comes from the assumption that a single-port block RAM is used between the particle filter stages [7]. Due to the nature of our proposal function, we require fewer particles ($100 < N \leq 1000$) than for the case when only the state update is used for particle proposal. Based on a clock frequency of 100 MHz, the latency or update rate in the DOA-only tracker for $N = 200$ is 9 μs and for $N = 1000$ is 33 μs. For $N = 200$ the overall latency is comparable to that of a bootstrap bearings-only tracker [12] that requires a large number of particles. This update rate of 9 μs to evaluate a single particle weight is sufficient to generate state estimates for each batch period $T = 1$ s.

6. REFERENCES

[1] C.-Y. Chong and S. P. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, Aug. 2003.

[2] V. Cevher and J. H. McClellan, "An acoustic multiple target tracker," in *IEEE SSP*, Bordeaux, FR, July 2005, pp. 17–20.

[3] V. Cevher, R. Velmurugan, and J. H. McClellan, "Acoustic multitarget tracking using direction-of-arrival batches," *IEEE Trans. Signal Processing*, vol. 55, no. 6, pp. 2810–2825, June 2007.

[4] ——, "A range-only multiple target particle filter tracker," in *Proc. IEEE ICASSP*, May 2006.

[5] V. Cevher, M. Borkar, and J. H. McClellan, "A joint radar-acoustic particle filter tracker with acoustic propagation delay compensation," in *Proc. 14th EUSIPCO*, Sept. 2006.

[6] M. Bolic, "Architectures for efficient implementation of particle filters," Ph.D. dissertation, Stony Brook University, New York, Aug. 2004.

[7] A. Athalye, M. Bolic, S. Hong, and P. M. Djuric, "Generic hardware architectures for sampling and resampling in particle filters," *EURASIP J. Applied Signal Processing*, vol. 17, pp. 2888–2902, 2005.

[8] *Xilinx System Generator for DSP v8.1 User Guide*, Xilnx, San Jose, CA, April 2006.

[9] M. Isard and A. Blake, "CONDENSATION – conditional density propagation for visual tracking," *Int. J. Computer Vision*, vol. 29, no. 1, pp. 5–28, Aug. 1998.

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

[11] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*. Chapman Hall/CRC, 2004.

[12] R. Velmurugan, "Implementation strategies for particle filter based target tracking," Ph.D. dissertation, Georgia Tech, Atlanta, GA, May 2007.

[13] J.-P. Deschamps, G. J. A. Bioul, and G. D. Sutter, *Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems*. Wiley-Interscience, 2006.

[14] *Xilinx University Program Virtex-II Pro Development System*, Xilinx, San Jose, CA, May 2006.

[15] *Embedded System Tools Reference Manual - Embedded Development Kit EDK 8.1i*, Xilinx, San Jose, CA, Oct. 2005, Xilinx ID: UG111 (v5.0).

[16] T. R. Halfhill, "Xilinx Revs up MicroBlaze," in *Microprocessor Report*, no. 01, Nov. 2006, pp. 13–17.