Local Decisions and Triggering Mechanisms for Adaptive Fault-Tolerance

Phillip Stanley-Marbell, Diana Marculescu
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890
{pstanley, dianam}@ece.cmu.edu

Abstract

Dynamic fault-tolerance management (DFTM) was previously introduced as a means of providing environment- and workload-driven adaptation for failure-prone battery powered systems.

This paper introduces and analyzes the role of local decision policies in a DFTM environment, and presents a precise formulation for when it is beneficial to activate a given DFTM algorithm with respect to metrics that combine performance, reliability, power consumption and battery life. In particular, local decision algorithms are described in the context of an imaging array application running on a network of resource-constrained processing elements. It is demonstrated that DFTM algorithms, in conjunction with appropriately chosen activation times, increase the mean computation before battery failure for a single battery, by a factor between 1.1 to 5.8, for the application investigated.

1. Introduction

Failure, in its many forms, is becoming an increasingly important design constraint in computing systems. Unlike traditional fault-tolerant computing systems, where failure is the exceptional case that must be addressed with appropriate preventive and recovery techniques, technology trends are making it increasingly difficult to guarantee device operability, with the result that larger fractions of manufactured devices must be declared defective when they fail testing. These factors make it increasingly important to include *reliability as a design constraint*, in much the same manner as power consumption has, in recent years, gained importance as a critical constraint alongside performance. By including failure as a design constraint, and characterizing systems in terms of a combination of performance, reliability, power consumption and where appropriate, battery life, it will be possible to derive greater utility from devices that would otherwise have to be discarded.

Failures in a system may be the result of manufacture-time defects, or may be the manifestation of runtime effects such as depletion of battery resources. In this regard, traditional power management may be considered a subset of the more general idea of *dynamic adaptation of a system to failures*. Low battery levels can then be seen as a predictable failure event, whose occurrence must be delayed as much as possible by taking appropriate actions, embodied by traditional power management algorithms. However, in the general case of failures, as is the case when final battery resource depletion does occur, fault free devices must be used as replacements for failing or failed devices. This requires the existence of such replacement devices or “spares” in a system, to be harnessed on the occurrence, or on the verge, of a failure.

Dynamic fault-tolerance management (DFTM) [8] was proposed as a framework in which dynamic adaptation to failures in the presence of redundantly deployed devices could be performed. It consists of three classes of algorithms which address adaptation with regard to local actions (subsuming traditional power management algorithms), redundantly deployed devices, and the choice of which of these spare devices should be utilized, respectively.

Contributions

Dynamic fault-tolerance management [8], as previously proposed, provided neither an investigation of DFTM policies which adapt the behavior of a device in a system without taking advantage of other redundantly deployed resources, nor a precise formulation for when a specific DFTM algorithm should be activated.

This paper presents an investigation of a set of DFTM *L-class* algorithms, and presents a precise formulation for when it is beneficial to activate a given DFTM algorithm, with respect to metrics which combine performance, reliability, power consumption and battery life. As previously introduced [8], *L-class* algorithms control local adaptation decisions, *M-class* algorithms control decisions of when to employ redundantly deployed resources, and *D-class* algorithms control the choice of which redundantly deployed devices to be harnessed. The results of this paper are applicable to systems which must adapt to the presence of

1530-1591/04 $20.00 (c) 2004 IEEE

failures such as mechanical failures or energy resource depletion, and must do so while providing maximal battery lifetime and sustained performance. Such system requirements are embodied by many emerging technology platforms, such as ambient intelligent systems [1] and integrated computational sensing and actuation surfaces.

The remainder of this paper is organized as follows. The following section presents motivation for performing dynamic adaptation to failures in the context of a driver application, a network of embedded processing elements acting as an imaging array. It is followed in Section 3 by a review of previously proposed measures [8] for quantifying the efficacy of a system in terms of a combination of **e**nergy-efficiency, **b**attery life, per**forma**nce and relia**bility**, known as *ebformability* measures [8]. A precise formulation for determining the conditions under which an algorithm should be activated, is presented in Section 4, followed by an experimental evaluation of the driver application in the context of the presented ideas, in Section 5. The paper concludes with a summary of contributions in Section 6.

2. Driver application : Imaging array

The driver application employed was one in which computing devices arranged in a 2-dimensional grid, sample values from sensors, and send these samples via a multi-hop network to a designated member of the grid. Such a configuration of devices is representative of several applications of relevance in ambient intelligent systems, such as an ultrasound imaging array for detecting the position, shape and motion of objects, or an acoustic beamformer. It is also representative of an active antenna array, used, for example, to provide spatial filtering for an ultra-wideband radio.

Many of the issues raised by the application are applicable to any system utilizing multiple resource-constrained processing elements that communicate via a multi-hop network. In the case of an ultrasound imaging array, the samples collected at each device in the network will contain information about the distance of objects from the particular sensor, and the final image (the collection of samples from all nodes in the system) will represent a 2-dimensional view of the distance of objects from the sensor array.

For each sampling period, the image sub-component captured at a device must be transmitted to whichever device in the system has designated itself as the aggregation point. The arrival times of the captured sub-components at the aggregation point will vary based on the properties of the communication network connecting nodes together, the method for routing data, as well as on the properties of the nodes. For example, in a network topology such as the grid shown in Figure 1, nodes in the system must cooperate in the routing of data, to enable non-adjacent nodes to communicate.

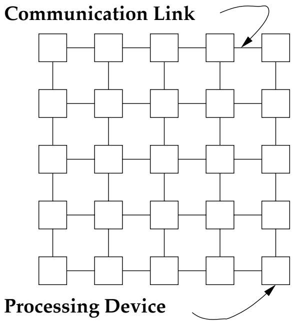


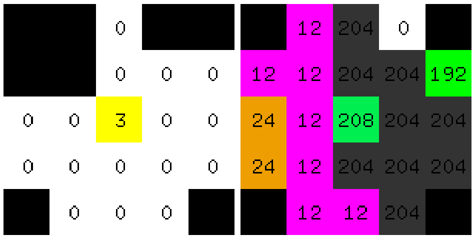
Figure 1. A grid topology with 25 nodes. Non-adjacent neighbor communication requires nodes to cooperate in routing.

The particular communication strategies employed at each node in the grid affect the communication latency between devices. For example, the algorithm governing the delay between transmission attempts when a communication link is busy (e.g., in the case of a CSMA/CD communication medium) will affect the communication latency, and will also affect the power dissipated at a device. In order to conserve energy, a node might choose not to perform forwarding for some or all received communication frames, and this decision might change over time. For example, in the event of impending system failure due to depletion of battery resources, a node might cease to forward a fraction of received communications. As a further example, in the presence of excessive communication failures (e.g., due to a physical fault in a communication link), a node might cease forwarding a fraction of received data to reduce energy wasted in retransmission attempts[1].

The variation in array sub-component arrival times or *coherence*, provides an intuitive measure of the quality of a set of samples obtained during a quantum of the system's lifetime. Figure 2 shows coherence maps for a 25-node system which employs an *as soon as possible (***asap***)* transmission back-off algorithm (a device will wait for the length of time taken to transmit one frame, and immediately retry transmitting, on the occurrence of a collision). The analogous coherence maps for a system employing a *random delay (***random***)* transmission back-off algorithm (wait for a random multiple of frame transmission times before retrying) are illustrated in Figure 3. The figures provide some intuition on the effects of one algorithm versus another on the system performance (and reliability, if, e.g., an image with greater than a threshold in standard deviation of arrival times must be discarded).

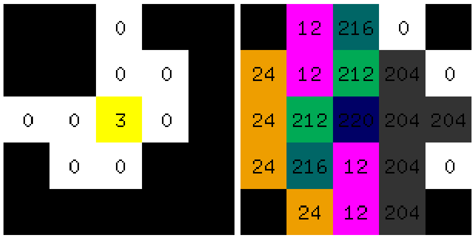
In order to judge the relative benefits of one DFTM al-

[1] The actual routing algorithm employed in the application investigated will not be described in detail here. It employs an initial topology discovery phase, followed by a deterministic routing strategy.



(a) Best σ_{img} (b) Worst σ_{img}

Figure 2. Spatial map of deviations in image subcomponent arrival times (in ms), for baseline asap retransmission delay. σ_{img} is the standard deviation in arrival times for the image components. Black squares denote absent samples.



(a) Best σ_{img} (b) Worst σ_{img}

Figure 3. Spatial map of deviations in image subcomponent arrival times (in ms), for baseline random retransmission delay. σ_{img} is the standard deviation in arrival times for the image components. Black squares denote absent samples.

gorithm versus another, it is necessary to employ metrics which include a combination of the effects of performance, reliability, power consumption and battery lifetime. Performance must be maintained for the obvious reasons of utility. Reliability is of increasing concern in systems which are either susceptible to high failure rates, or which employ failure as a means of "throttling" performance (e.g., dropping frames to be forwarded in order to conserve energy). Power consumption and battery life must be treated separately, since in some cases, a longer battery life might be achieved at the cost of an increased peak power dissipation, which might lead to increased heat dissipation and possibly to reduced reliability. The following section provides a more thorough formalization of what it means for one system to be "better" than another, in terms of a combination of measures of performance, reliability, power consumption and battery life.

3. Combined metrics

3.1. Overview

Failure of a subset of the system resources can be tolerated in gracefully degrading fault-tolerant systems [3], possibly resulting in degraded performance. The redundantly deployed devices which help maintain system functionality in the presence of failures, provide additional performance in their absence. In such situations, it is therefore necessary to employ metrics that take into account both system performance and reliability [2]. For systems in which performance, power (average and peak power, and overall energy consumption) and battery lifetime may be traded-off for reliability, similar measures are required.

In the following, F is the set of failure states, a subset of the states in which a system may exist, which are indexed with the variable i. The initial system state is always denoted by I. The probability of being in a state i after n time steps is denoted by $P_i(n)$. The behavior of a system under study can be characterized by a collection of distinct states, each state corresponding to a given level of performance. For example, in a gracefully degrading system with N nodes, 3 or more of which must be functioning in order for the system to be considered "alive," the system may be modeled as a set of $N+1$ states, of which states 0, 1 and 2 constitute the set of failing states.

The amount of computation that can be performed by a system, T, the *computation availability* [2], is defined as:

$$T = \alpha \cdot n \tag{1}$$

where α is the *computation capacity*—the amount of useful computation per unit of time, and n is the number of time steps.

$C_i(T)$, the capacity function [2], is the probability that the system executes a task of length T before its first failure, given that the state at the start of computation was i:

$$C_i(T) = \sum_{j \notin F} P_j^*(T) \tag{2}$$

$P_j^*(T)$ (or $P_j(n)$ for $T = \alpha \cdot n$) is the probability of being in a given state after $T = \alpha \cdot n$ amount of computation [2]. Larger values of $C_i(T)$ are desirable for a given T.

In a battery powered system, the variation of $C_i(T)$ will be affected by the battery discharge profile, and will be bounded by the battery life. The *battery-aware capacity function* [8], $Cbatt_i(T)$, is defined as:

$$Cbatt_i(T) = \sum_{j \notin F} P_j^*(T) \cdot \zeta_{batt}(T) \tag{3}$$

where $\zeta_{batt}(T)$ is obtained by transforming the variation of the battery state of charge versus time curve, (which can be derived from the data-sheet for a particular battery cell), into the computation domain. In its simplest form, it is a step function whose extent is determined by the battery lifetime.

$C_i(T)$ and $Cbatt_i(T)$ can be used to calculate the *Mean Computation Before Failure (MCBF)* [2] and *Mean Computation Before Battery Failure (MCBBF)* [8] respectively:

$$MCBF = \sum_{T=0}^{T=\infty} C_I(T), \tag{4}$$

$$MCBBF = \sum_{T=0}^{T=\infty} Cbatt_I(T) \tag{5}$$

where I is the initial state, at $n = 0$ (and $T = 0$).

The above measures can be used to determine the efficacy of a system in providing fault-tolerant operation with the best combination of power consumption, performance, reliability and longest battery lifetime. Such an evaluation would proceed by first defining the system states, and determining the transition probabilities between states. The probability of being in a given state, i, after a number of time steps, $P_i(n)$, can then be obtained by any of the methods described in the literature [4]. $P_i(n)$ can then be transformed to $P_i^*(T)$, by expressing the probability of being in a given state, in terms of the amount of computation performed, rather than time steps. This requires an appropriate definition, for the particular system under study, of the computation capacity, α. The metrics taking into account performance, reliability, battery life and power consumption can then be determined as detailed in equations 1–5 above. These metrics, which combine the notions of energy-efficiency, battery lifetime, performance, and reliability, are referred to as *ebformability* measures [8].

3.2. Parameters for image array application

For the image array application, a failure is defined as the arrival time of a sample at the aggregation node being more than one standard deviation from the mean for that sampling period. The states of the system are defined to correspond to the number of valid samples received.

The computation capacity [2], α, is the amount of useful computation performed per time step. It embodies the "quality" of the behavior of the system. For the application under study, four factors must be taken into consideration regarding performance: (1) The standard deviation, σ_{img}, between the arrival times of components of an image, averaged across many images; (2) The CPU occupancy, occ_{cpu}, the fraction of time for which the CPU is busy; (3) The maximum occupancy of the transmit and receive FIFO queues, occ_{fifo}, across all the nodes in the system and (4) The average number of valid image subcomponents received, $nvalid$. The CPU occupancy and maximum buffer lengths used, are included in the measure of computation capacity to incorporate a notion of how efficiently resources are being used. Given that the application was employed on a non-preemptive system executing a single application, they provide a measure of the additional load that the system could bear, in the presence of preemptive multiprogramming. The computation capacity, α is therefore defined as a combination of these factors:

$$\alpha = \left(\frac{\epsilon}{occ_{cpu}} + \frac{\mu}{\sigma_{img}} + \frac{\nu}{occ_{fifo}} \right) \cdot nvalid \tag{6}$$

In the above equation, ϵ, μ and ν are the relative weights attached to the roles of occ_{cpu}, σ_{img} and occ_{fifo}, respectively. In the experimental evaluation, identical weighting factors were employed for all the terms, i.e., $\epsilon = \mu = \nu$.

4. DFTM algorithm triggering mechanisms

Dynamic Fault-Tolerance Management (DFTM) involves dynamically adapting systems in the presence of failures, by employing a structured set of algorithms which modify a node's behavior or take advantage of alternate resources, to provide prolonged lifetime. The failures in question may be *intermittent non-fatal* failures (such as communication errors) or *predictable fatal* failures (such as low battery levels). Despite the usefulness of the structured approach to adaptation algorithms, the benefits to be derived from algorithm activations is dependent on the time at which these activations occur. The original proposal of DFTM [8] used educated guesswork to determine the threshold at which the activation of a DFTM algorithm should occur. It is however possible to derive precise formulations for when an algorithm should be activated, given the tradeoffs between the overheads associated with the algorithm, in terms of a combination of performance, power, reliability and battery life, and its benefits thereof.

In order to achieve the best combination of performance, reliability and battery lifetime for a battery-powered system, it is desirable for a DFTM algorithm to maximize the *Mean Computation Before Battery Failure (MCBBF)* ebformability measure. The battery threshold for initiating the action of a DFTM algorithm (e.g., switching from the **asap** collision retry policy to the **random** collision retry policy for the example in Section 2), henceforth referred to as the *activation threshold*, $\xi(Battery, Algorithm, \gamma)$, is given as:

$$\xi(Battery, Algorithm, \gamma) = \frac{\theta_{alg}(Battery, \gamma)}{lifetime_{base}} \tag{7}$$

where θ_{alg} is the time of activation of the new behavior, $lifetime_{base}$ is the expected lifetime of the system if the algorithm is not activated, and $0 \leq \gamma \leq 1$, is the relative importance of the combined ebformability metrics over optimal system lifetime. The constant γ enables a specification

Number of nodes	25 (26 for M-class algorithms)
Link Layer Frame Size	1024 bits
Link Speed	1 Mb/s
Communication Power	250mW TX/RX
Operating Frequency	15 MHz
Battery Capacity	0.1 mAh
Battery Discharge Prop. LUT	Panasonic CGR18
DC-DC Conv. Efficiency LUT	Maxim MAX1653

Table 1. Properties of the simulated system

of the emphasis to be placed on the ebformability measures versus on achieving the optimal system lifetime (without regard, say, to reliability).

In activating a new behavior over the baseline operation, the time of activation affects the overall lifetime in two ways. The obvious effect is any change in performance from the activation of the new behavior. However, the time at which the activation occurs, also affects the usable lifetime of the battery cell, as a result of non-linearities in the battery electro-chemical characteristics, particularly, if the activation might cause a temporary or permanent change in the current discharge profile. The fractional increase in battery lifetime derived from employing an activation time of θ_{alg} is λ_{alg}, a function of θ_{alg}, $lifetime_{base}$ and the battery cell characteristics. It can be determined using methods for determining the effect of application profile on battery lifetime [5].

The goal in obtaining θ_{alg} for the algorithm alg, is to maximize the increase in MCBBF over a baseline, $base$, given knowledge of the MCBBFs for $base$ and alg, and the discharge characteristics of the battery cell, i.e.:

$$\begin{aligned}\Delta \quad = \quad & \sum_{T=0}^{\alpha_{base}\cdot\theta_{alg}} C_{base}(T)\zeta_{batt}(T) \\ & + \sum_{T=\alpha_{base}\cdot\theta_{alg}}^{\alpha_{alg}\cdot lifetime_{base}\cdot\lambda_{alg}} C_{alg}(T)\zeta_{batt}(T) \\ & - \sum_{T=0}^{\alpha_{base}\cdot lifetime_{base}} C_{base}(T)\zeta_{batt}(T) \geq 0 \quad (8)\end{aligned}$$

where T, α, n and $C_i(T)$ are as previously defined in equations 1–5.

Equation 8 can be solved to obtain the θ_{alg} for which there is an increase in the MCBBF, and the optimal θ_{alg} can be obtained by taking the appropriate derivatives of Δ.

5. Evaluation

The imaging array application was implemented in the C programming language, and compiled for execution on a cycle-accurate simulator, an extension of a previously developed simulation framework [7], that enables the instantiation of multiple processing elements, modeling computation, communication and battery subsystem properties[6]. From the simulation data, the parameters occ_{cpu}, occ_{fifo} and σ_{img} of Equation 6 are extracted, as well as the resultant battery lifetime and transition probabilities between states. The properties of the simulated system are listed in Table 1, and the set of DFTM algorithms investigated are listed in Table 2.

Algorithm	**Description**
binexp	Binary exponential transmission retry delay (**binexp**).
asap	ASAP transmission retry delay (**asap**).
random	Random transmission retry delay (**random**).
dropfwd	Randomly drop 10% of link layer frames meant to be forwarded (**dropfwd**).
L0	Switch from **binexp** to **asap** after (ξ(# collisions) = 32) link layer collisions.
L1	Switch from **binexp** to **random** after (ξ(# collisions) = 32) link layer collisions.
L2	Enable **dropfwd** after occurrence of (ξ(# collisions) = 32) link layer collisions.
L3	Enable **dropfwd** after (ξ(batt. level) = 25% capacity).
L4	Switch from **binexp** to **asap** after (ξ(batt. level) = 25% capacity).
L5	Switch from **binexp** to **random** after (ξ(batt. level) = 25% capacity).
L6	Enable **dropfwd** after occurrence of (ξ(batt. level) = 15% capacity).
M0	Re-map application sub-component to a redundantly deployed device after (ξ(batt. level) = 25% capacity).
M1	Re-map application sub-component to a redundantly deployed device after (ξ(batt. level) = 15% capacity).

Table 2. Baseline, L-class and M-class DFTM instantiations investigated.

The first group of system configurations listed in Table 2, **binexp** to **dropfwd**, represent the baseline configurations, with the strategy listed in the second column employed throughout the application's execution (i.e., no dynamic changes between behaviors in response to application or environment conditions). The second group, L0 to L6 represent seven simple DFTM L-class algorithms for adaptation. In the second group, ξ represents some function of the number of link layer collisions, or the battery capacity, at which the algorithm's change should occur, as shown in Equation 7. The last group represents DFTM M-class algorithms, which employ application re-mapping to take advantage of redundantly deployed devices in a system. These employed the same threshold as the corresponding L-class algorithms, to investigate the possible benefits of prioritizing algorithms from the different classes at a given threshold.

The trend in the mean computation before battery failure (MCBBF) across the baseline, L- and M-class algorithms is shown in Figure 4. The MCBBF of a system represents the average amount of computation the given sys-

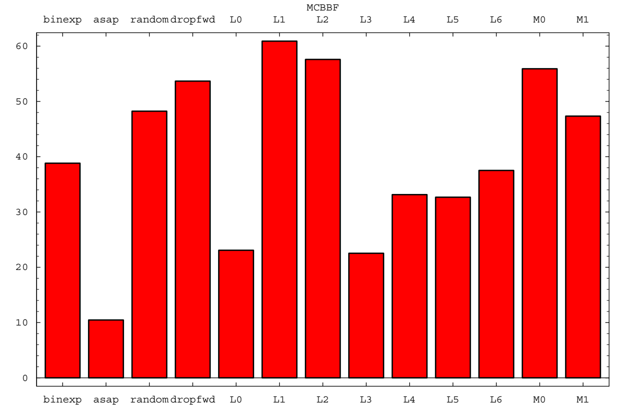


Figure 4. Mean Computation Before Battery Failure, MCBBF.

tem configuration will complete, in the presence of possible system failures, before it exhausts its battery. Even though the **asap** baseline witnesses the smallest average communication buffer occupancy, implying the best communication performance, it does not achieve the lowest standard deviation in image component arrival times (the data is not presented here for brevity). It also achieves the worst MCBBF of all the baseline configurations. This is because even though the individual indicators of performance (CPU occupancy, standard deviation across image subcomponent arrival times, average buffer occupancy) are comparable or better than the other baseline configurations, the probability of being in a non-failed state, with increasing amount of computation performed, as given by $C(T)$ (see Section 3), is substantially smaller than for the other baselines.

The best baseline MCBBF is achieved by the **dropfwd** algorithm, which selectively drops 10% of the received packets meant to be forwarded. Even though on the average it has an approximately 50% larger standard deviation in image subcomponent arrival times, on the whole, it exhibits a greater probability of being in a non-failed state over time. This best performing baseline is improved on by the L1 and L2 L-class algorithms, and the M0 M-class algorithm. The L1 algorithm switches from a **binexp** retransmission delay to a **random** delay after a threshold of 32 link layer collisions. It witnesses a 1.1x improvement over the best performing baseline MCBBF (**dropfwd**), and a 5.8x improvement over the worst baseline MCBBF (**asap**).

Comparing L- and M-class policies which are enabled at the same battery capacity threshold (L3, L4 and L5 compared to M0; L6 compared to M1), it is observed that in terms of the MCBBF, for the same activation threshold, the M-class algorithms should be given priority over L-class algorithms, if the goal is to maximize MCBBF. For different battery sizes and characteristics however, the benefits from employing L-class algorithms might begin to exceed those of M-class algorithms for the same threshold. An interesting avenue for future research is determining the threshold in average battery capacity in a system, for which M-class algorithms will always perform better than L-class algorithms activated at the same threshold, and vice versa.

6. Summary

This paper presented an investigation of a set of DFTM *L-class* algorithms, and a precise formulation for when it is beneficial to activate a given DFTM algorithm, with respect to metrics which combine performance, reliability, power consumption and battery life. It was demonstrated with the aid of a concrete example and detailed simulation results, that L-class algorithms can provide a substantial improvement over systems not employing DFTM. The mean computation before battery failure for a single battery was shown to be increased by a factor between 1.1x to 5.8x, for the application investigated.

Acknowledgments

This research was supported in part by the Army Research Office through the Center for Computer and Communication Security at Carnegie Mellon University and by the Semiconductor Research Corporation under Grant No. 2002-RJ-1052G.

References

[1] T. Basten, L. Benini, A. Chandrakasan, M. Lindwer, J. Liu, R. Min, and F. Zhao. Scaling into Ambient Intelligence. In *Proceedings of Design Automation and Test in Europe, DATE'03*, pages 76–81, March 2003.

[2] M. D. Beaudry. Performance-related reliability measures for computing systems. *IEEE Transactions on Computers*, c-27(6):540–547, June 1978.

[3] B. R. Borgerson and R. F. Freitas. A reliability model for gracefully degrading and standby-sparing systems. *IEEE Transactions on Computers*, c-24:517–525, May 1975.

[4] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, 2001.

[5] D. Rakhmatov, S. Vrudhula, and D. A. Wallach. Battery Lifetime Prediction for Energy-Aware Computing. In *International Symposium on Low Power Electronics and Design, ISLPED'02*, pages 154–159, August 2002.

[6] P. Stanley-Marbell. Myrmigki Simulator Reference Manual. Technical report, CSSI, Dept. of ECE, Carnegie Mellon, 2003.

[7] P. Stanley-Marbell and M. Hsiao. Fast, Flexible, Cycle-Accurate Energy Estimation. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 141–146, August 2001.

[8] P. Stanley-Marbell and D. Marculescu. Dynamic Fault-Tolerance and Metrics for Battery Powered, Failure-Prone Systems. In *International Conference on Computer Aided Design, (ICCAD)*, pages 633–640, November 2003.