Online Adaptation Policy Design for Grid Sensor Networks with Reconfigurable Embedded Nodes

Varun Subramanian, Michael Gilberti and Alex Doboli
Department of Electrical and Computer Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794–2350
Email: adoboli@ece.sunysb.edu

***Abstract*—This paper presents a systematic methodology for designing the adaptation policies of reconfigurable sensor networks. The work is motivated by the need to provide efficient sensing, processing, and networking capabilities under tight hardware, bandwidth, and energy constraints. The design flow includes two main steps: generation of alternative design points representing different performance-cost trade-offs, and finding the switching rates between the points to achieve effective adaptation. Experiments studied the scaling of the methods with the size of the networks, and the effectiveness of the produced policies with respect to data loss, latency, power consumption, and buffer space.**

I. Introduction

Networked embedded sensor systems (sensor networks) are rapidly emerging as a key concept for many modern applications, like environmental monitoring, energy conservation, healthcare, infrastructure management, and so on [1], [2]. This is possible because many sensing and electronic devices are small and cheap, thus, deployable in large numbers. Sensor networks offer unique benefits, such as significantly better data acquisition and decision making over broad physical regions. Reliability and robustness are also much improved as failures of local nodes do not disrupt the functioning of the entire networked system.

Typical applications, e.g., monitoring and tracking, define for a variety of situations the global tasks and goals that must be achieved by a sensor network through combined operation of its embedded nodes. As a result, two main design challenges emerge: (i) each sensor node has to efficiently sense, process, and network under a wide range of performance requirements while (ii) only scarce hardware, bandwidth, and energy resources are available (to keep the cost low). Present design methods are insufficient for tackling the two challenges. Methods have only limited capability for co-optimizing the sensing, processing, and communication subsystems of embedded nodes. Also, few methods exploit the flexibility of networked reconfigurable architectures to produce low-cost yet efficient designs for a broad range of requirements.

The sensor network research community has traditionally studied network routing protocols [1] without focusing much on design methods for networks of reconfigurable sensor nodes. Fortunately, there is some similarity between sensor networks and Networks-on-Chip (NoCs) [3], [4], [5]. For both, the performance of the networking infrastructure is critical for the overall system performance. Algorithms for NoC path selection, mapping of cores, and TDMA time slot allocation are discussed in [6]. Methods for communication topology mapping, and topology and protocol selection are offered in [7], [4], [8], [9]. Ascia *et al.* [7] present a Genetic Algorithm for finding the communication topology, protocols, and priorities of mesh-based NoCs with static routing of communication packets. Bertozzi *et al.* [4] suggest a communication network design flow that includes topology mapping, selection, and generation for minimizing area, power, hop delay, and bandwidth of various topologies, like mesh, torus, hypercube, etc. Murali *et al.* [8] propose a methodology to map different use-cases to reconfigurable NoCs, so that the constraints of each use-case is met. Guz *et al.* [10] discuss allocation of non-uniform link capacities under QoS timing constraints. The related work on NoCs offers very interesting methods for designing the networking infrastructure, but does not tackle the interdependency between the design of reconfigurable nodes and the network. Exploiting the adaptation opportunities of networked embedded nodes reduces data loss, latency, and power consumption without increasing cost.

This paper proposes a systematic procedure for designing the adaptation policies of reconfigurable sensor networks. The policies control online the characteristics of the sensor nodes and data routing so that the performance requirements of the network are optimized, such as data loss, power consumption, and latency. The method also considers any constraints imposed by the application, like minimum data sampling rate, and hardware resources, e.g., local memory size. The proposed procedure includes two steps: (i) Design Point (DP) generation, and (ii) adaptation policy design. The first step produces DPs that represent multiple performance - cost trade-offs. Using a Linear Programming solver, the second step calculates the switching rates between alternative DPs and possible communication paths, so that the requirements are optimized. The paper also details the formal models used by

978-3-9810801-5-5/DATE09 © 2009 EDAA

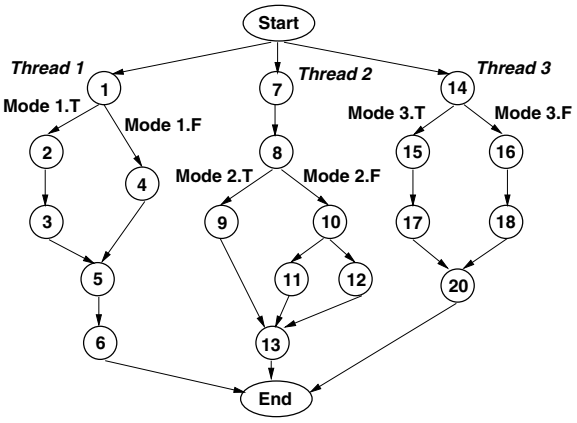

Fig. 1. Multimode Dataflow Graph

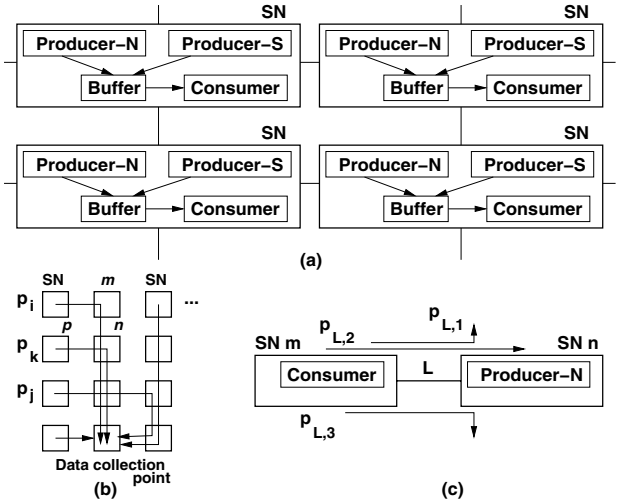

Fig. 2. Data Communication Network

the two design steps: Multimode Dataflow Graphs are used by DP generation to capture the multiple operation modes of sensor nodes. Data Communication Networks, based on Continuous-Time Markov Chains [11], offer an aggregated description of the data flow through a sensor network.

The work puts forward several novel concepts. It offers a method for integrated co-optimization of sensing, processing, and networking in networks of embedded nodes. Also, it proposes scalable techniques to systematically design the adaptation policies of networks of reconfigurable nodes.

The paper has the following structure. Section 2 presents the formal descriptions used in design. Section 3 discusses the proposed methods for SN adaptation. Section 4 offers experimental results, and Section 5 brings conclusions.

II. Formal Models for Adaptation

The proposed design flow is as follows. Specifications are compiled by a server into executable code for the reconfigurable nodes of a sensor network. Code is produced for N Design Points (DPs) representing different performance tradeoffs. The code is transmitted to all nodes. The execution manager of each node controls the node's adaptive behavior by switching among the N DPs depending on the sensor inputs, performance requirements, and events. This section presents the formal models used for finding the adaptation policies of the execution manager.

The used specification notation for wide-area, distributed data acquisition is based on the concept of *Data Blankets* (DBs) [12]. A DB corresponds to a geographical area. Data is continuously sampled from the area, aggregated, and then used in decision making or control. In addition to its associated physical area, each DB includes a *Description Module* (DM) with five components: (i) sensed inputs, (ii) control outputs, (iii) capabilities and constraints of the embedded architectures and network (e.g., communication bandwidth, memory, energy, sensing resolution, etc.), (iv) application goals (like latency, power consumption, maximum data loss, etc.), and (v) the algorithms for the functionality of the nodes.

Two representations are used in compiling DB specifications into code for the node's reconfigurable hardware: (A) Multimode Dataflow Graphs (MDGs) describe the multi-mode functionality of the nodes, and (B) Data Communication Networks (DCNs) express the overall data communication behavior of the network. As Section 3 explains, MDGs are used to produce alternative Design Points (DPs) for the reconfigurable nodes, and DCNs are utilized in finding the adaptation policies for switching among DPs.

A. Multimode Dataflow Graph (MDG). MDGs are polar graphs that consist of multiple parallel threads spanning between the start and end nodes, as shown in Figure 1. Each thread is composed of nodes connected through arcs. A node describes an algorithmic component, e.g., function, subroutine, etc. Arcs express the required sequencing and data dependencies between nodes. Nodes communicate through shared memory. Nodes can share hardware resources for their implementation.

Each thread might include multiple modes. A mode represents a pair of polar sub-graphs, which are executed in mutual exclusiveness depending on the value of the condition associated to the mode. Condition values change continuously. For example, in Figure 1, Nodes 2 and 3, and Node 4 define the two complementary branches of Mode 1. The first branch is selected if the condition is true (labeled $Mode1.T$), and the second branch if the condition is false (labeled $Mode1.F$). Modes can be nested.

B. Data Communication Network (DCN). DCNs offer an aggregated description of the data flow through the entire network. DCNs are a network of Sensing Nodes (SNs), as shown in Figure 2(a). The DCN topology defines the interconnection of SNs, e.g., mesh, torus, hypercube, etc. This paper considers only grid topology even though the methodology supports other topologies also.

DCNs give two perspectives: (i) the local, SN-level production and consumption of data, and (ii) the network-level data flow between SNs. The two issues are discussed next.

i. Local production and consumption of data. At the SN-level, data flow is modeled as producers that place data

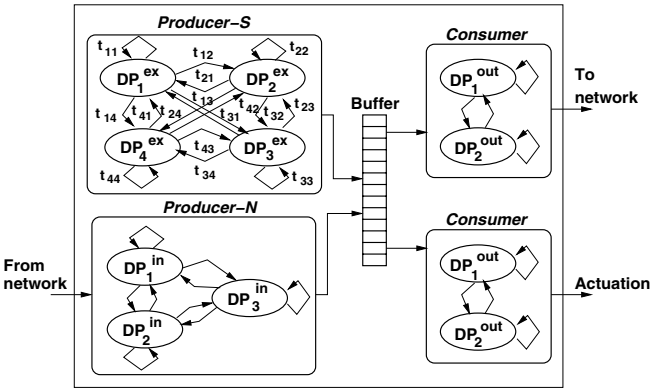

Fig. 3. Producer-consumer model of SNs

into a common buffer, and consumers that remove data from the buffer. The buffers represents the local memory of an embedded sensor node. Moreover, producers are of two kinds: *Producers-S* represent data sensing through the physical sensors, and *Producers-N* correspond to data input from the network. Consumers model the data output to the network as well as any local data processing, e.g., filtering, local data aggregation, etc. Consumers remove data from the buffer without adding data back to it. Figure 3 illustrates the producer-consumer model.

Each producer and consumer corresponds to one parallel thread of the MDG. For example, the thread for temperature sensing has a related Producer-S in the DCN. Similarly, the threads for data input (output) from (to) the network have corresponding Producers-N and Consumers in the model. A thread can have different data production (consumption) characteristics, e.g., production rate, data precision, etc., depending on the selected modes and implementation. Each implementation of a thread is called Design Point (DP). Hence, a producer (consumer) is described by a set of distinct DPs. Moreover, during operation, producers and consumers switch among their DPs to adapt to the specific conditions.

Similar to [11], the sensor node-level data production and consumption is described as a Continuous-Time Markov Chain (CTMC):

$$Producer_I(Consumer_I) = (S_I, A, A(i), t, K, r) \qquad (1)$$

For producer (consumer) I, S_I is the set of states, A is the action set, and $A(i)$ are the actions associated to state $i \in S_I$. $t(i,j,a)$ is the transition rate if action a is chosen for transitioning from state i to state j. K is the number of reward criteria. $r_k(i,a)$ is the reward rate if action a is selected for state i. α_i is the steady-state probability of state i.

For example, in Figure 3, *Producer-S* has four states, each state being a different design point DP_i. Similarly, *Producer-N* has three states. The reward criteria are the performance attributes of the DPs, such as latency, power consumption, data rate, etc. The action set A has only one action, which is defined by the fixed functionality of a DP. Hence, for brevity, actions are not indicated in the following CTMC equations.

The following equations describe each state i of producer

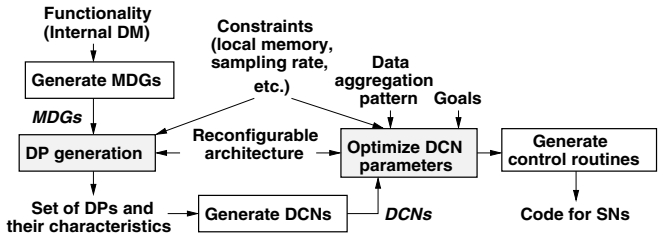

Fig. 4. Design flow for reconfigurable SNs

(consumer) I:

$$t(i,i)\ \alpha_i - \sum_{\forall k \in S_I} t(k,i)\ \alpha_k = 0 \qquad (2)$$

$$\sum_{\forall i \in S_I} \alpha_i = 1, \qquad \alpha_i \geq 0 \qquad (3)$$

Equation (2) correlates the rate of using DP (state) i to the rate of using DP k and the transition probability from k to i. Equation (3) indicates that one DP is always used for a thread.

Assuming that each state i of producer (consumer) I produces (consumes) data at rate q_i ($q_i \geq 0$) then the average amount of data produced (consumed) in one unit of time is:

$$DATA_I = \sum_{\forall i \in S_I} \alpha_i q_i, \qquad (4)$$

Finally, the amount of data generated by all producers of an SN has to be larger than the amount of data consumed by all its consumers:

$$\sum_{\forall I \in Producers} \sum_{\forall k \in S_I} DATA_k \geq \sum_{\forall J \in Consumers} \sum_{j \in S_J} DATA_j \qquad (5)$$

ii. Network-level data flow. The network-level data flow is modeled as a Data Aggregation Pattern (DAP). A DAP is a set of paths so that every SN is connected to a data collection point (target) in the network. An SN can use multiple paths. Figure 2(b) illustrates a DAP including paths p_i, p_k, p_j, and so on. SN n receives data from SN m through path p_i, and from SN p through path p_k. Patterns help the expressing of the communication parameters of connected SNs, e.g., communication load and delay.

The model satisfies the following data conservation rule: the data output by one SN is entirely received by the inputs of the SNs connected to it. Lossy connections can be modeled by transfer functions of links. Without affecting the generality of the method, the paper considers only lossless links. As shown in Figure 2(c), for link L between any SNs m and n, set P_L is the set of all paths $p_{L,k}$ that use link L. B_{p_k} is the bandwidth and $\beta_{p_{L,k}}$ is the probability of using path p_k. Using equation (4) and the data conservation rule, the following equation states that P_L should have sufficient bandwidth for the communication between SNs m and n:

$$\sum_{\forall i \in S_{consumer_m}} \alpha_i q_i = \sum_{\forall j \in S_{Prod-N_n}} \alpha_j q_j \leq \sum_{\forall p_{L,k} \in P_L} \beta_{p_{L,k}} B_{p_k} \qquad (6)$$

and,

$$\sum_{\forall p_{L,k} \in P_L} \beta_{p_{L,k}} = 1, \qquad \beta_{p_{L,k}} \geq 0 \qquad (7)$$

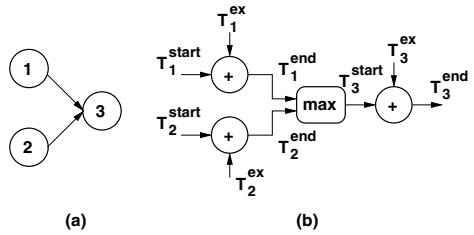

Fig. 5. PMs for scheduling

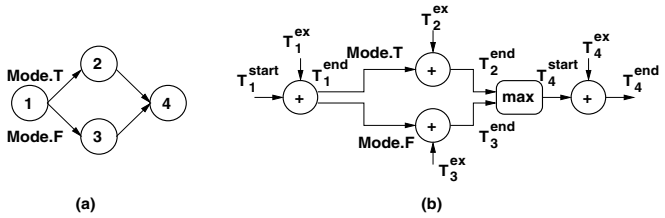

Fig. 6. Scheduling with multiple modes

III. Two-Step Design Flow

Figure 4 presents the proposed design flow. For a distributed data acquisition application, the flow computes the adaptation policies of the reconfigurable SNs. The flow includes two main steps: (i) Design Point generation and (ii) adaptation policy design. (i) MDGs compiled from the input specification (e.g., Internal DMs) are used to produce multiple DPs. DPs represent different performance-cost tradeoffs, and correspond to various operation modes (due to different values of the associated conditions), performance requirements, and hardware resources. (ii) Next, the produced DPs, the architectural characteristics of SNs, and the application goals are used to set-up the DCN model of the network. The DCN parameters, like switching rates between DPs and routing paths, are computed to optimize goals, like minimize the overall data loss, power consumption, and latency. The parameter values are used in the adaptation routines to control the switching between different implementations and communication paths.

This paper focuses on the design steps showed as shadowed boxes in Figure 4. The two steps are discussed next.

3.1 Design Point Generation. First, the design flow generates multiple design points (DPs) that correspond to different performance trade-offs, operation modes, and hardware sets. The method is based on the algorithm proposed in [9], which handles systems with multiple operation modes. The algorithm co-optimizes all modes together as opposed to traditional methods, which consider only one mode at a time, usually starting with the most performance-constrained mode [8]. As shown in [8], traditional strategies can result in hardware over-design as high as 90% compared to the optimal solutions.

The algorithm for generating multiple DPs is based on Simulated Annealing (SA). It finds the resource mapping and scheduling of MDG nodes for different hardware resource sets. The cost function is a weighted sum of the performance attributes. The exploration loop performs two types of moves: it changes with probability p the mapping of a node to resources, and it modifies with probability $(1-p)$ the order of execution of two nodes mapped to the same hardware resource. Hence, the algorithm conducts simultaneous mapping and scheduling, which offers better solutions than having two separate steps [9].

Performance Models (PMs) is the representation used by SA to perform simultaneous mapping and scheduling. A PM is an acyclic graph that corresponds to a given MDG, and is used to evaluate the performance attributes of an MDG's implementation, e.g., latency, power consumption, etc. PMs include variables and operator nodes. Variables denote the performance values of the blocks in the implementation. For example, in Figure 5(b), variable T_1^{ex} is the execution time of Node 1 for its current hardware binding (for example, the execution time of a task for certain resources). Similarly, variable T_2^{ex} is the execution time of Node 2, and so on. Variables T_i^{start} and T_i^{end} are the start and end times of Node i. Operator nodes correspond to operators and mathematic functions, e.g., plus, maximum, etc., and are used to compute the overall performance attributes of an implementation based on the node attributes and the specific scheduling order. For example, Figures 5, 6, and 7 are the PMs used to evaluate the overall latency of an implementation. Similar graphs can be set-up for other attributes, like power and energy consumption. An overall performance attribute is computed by pre-order traversal of the graph. Hence, performance estimation with PMs has module-level accuracy.

PMs offer the benefit of simplifying the combined expression of mapping and scheduling. Figure 5(b) illustrates the PM structure built to express the data dependencies between nodes 1, 2, and 3 in Figure 5(a). The PM graph shows that the start time T_3^{start} must be larger than both end times T_1^{end} and T_2^{end} of nodes 1 and 2. Moreover, for each node i, its end time T_i^{end} is the sum of its start time T_i^{start} and execution time T_i^{ex}. Figure 6(b) illustrates the PM sub-graph set-up for the two modes in Figure 6(a). The semantics of the PM is similar to the previous case with the difference that the branches for the unselected modes propagate the value $-\infty$ to the *max* operator node. Thus, the unselected modes do not affect the overall performance. If the hardware binding of a node changes then the value of the T^{ex} variable changes too. The scheduling order between two nodes mapped to the same resource is expressed through a dotted arc, as shown in Figure 7(b). If Node 2 in Figure 7(a) is executed before Node 1 then the dotted arc in the PM reflects the ordering constraint. The SA loop changes the execution order between the two nodes by modifying the direction of the dotted arc. A dotted arc is dynamically added whenever two nodes share the same resource, and is removed if the mapping for one of the nodes changes.

3.2 Adaptation Policy Design. This second step uses equations (2)-(7) in the DCG model to optimize the dynamic behavior of a SN network. The procedure calculates the steady-state probabilities and transition rates of DPs and communication paths to optimize a cost function that expresses the application

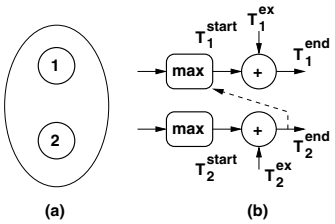

Fig. 7. Scheduling for resource sharing

goals, data loss, latency, and power consumption. In addition, the step considers the constraints of the architecture, such as available memory.

The data loss of node K per one unit of time is the difference between its overall production and consumption rates:

$$Loss_K = \sum_{\forall i \in S_{K,producers}} \alpha_i \ q_i - \sum_{\forall i \in S_{K,consumers}} \alpha_j \ q_j \quad (8)$$

The buffer capacity of any node K must be larger than the worst-case buffer space requirement:

$$\sum_{\forall i \in DecO(S_{K,prod.})} \alpha_i \ q_i - \sum_{\forall j \in IncO(K,S_{consumers})} \alpha_j \ q_j \leq Buff_K \quad (9)$$

Sets $DecO$ and $IncO$ are the set of states of the producers and the set of states of the consumers, respectively, for which data keeps accumulating in the internal buffer. In the worst case, input data is coming at the highest rate of the producer states while data is consumed at the lowest rates of the consumers. Hence, the states in sets $DecO$ and $IncO$ include the DPs with the top k largest input rates and the DPs with the l-lowest output rates, such that $\sum_{i \in DecO} \alpha_i \ q_i \geq \sum_{j \in IncO} \alpha_j \ q_j$. Set $DecO$ is ordered in the decreasing order of the rates, and set $IncO$ in the increasing order of the rates. Values k and l are the values for which the difference in equation (9) is maximum.

The average power consumption of the network is the total power consumption of all SN K in the network:

$$P^{total} = \sum_{\forall K} \sum_{\forall S_I \in K} \sum_{i \in S_I} \alpha_i \ P_i \quad (10)$$

The average input data rate (sensing precision) of SN K is:

$$Input\_rate_K = \sum_{j \in Producer - S_K} DATA_j \geq Resolution_k \quad (11)$$

The delay of path p_k is the average latency of all its SNs:

$$Delay_{p_k} = \sum_{\forall m \in p_k, I \in m} \sum_{i \in S_I} \alpha_i \ Delay_i \quad (12)$$

The latency of the network is equal to:

$$Latency_{aver} = \max_{p_k} Delay_{p_k} \quad (13)$$

The cost function minimizes a weighted sum describing the overall data loss, power consumption, and delay:

$$Cost = \gamma \ \max_i Loss_i + \delta \ P^{total} + \mu \ Latency_{aver} \quad (14)$$

Ex.	Best-case			Worst-case			Diff. (%)	Exec. time (sec)
	List sch.	Opt.	Imp. (%)	List sch.	Opt.	Imp. (%)		
SN 16	82	41	50	136	108	20	62	33
SN 24	154	84	45	210	180	14	53	34
SN 32	173	156	9.8	242	215	11	27	112
SN 48	522	296	43	510	353	30	16	97
SN 64	633	440	30	852	510	40	19	225.30

TABLE I
DESIGN POINT GENERATION

IV. EXPERIMENTS

The experimental part presents results for (i) DP generation and (ii) adaptation policy design. The section also discusses the design of adaptation policies for a grid sensor network with 27 PSoC embedded processors [13].

i. Design Point generation. The first set of experiments studied the quality of DP generation using the proposed method. MDGs with 16, 24, 32, 48, and 64 nodes have been considered. Each MDG had three modes. Five different resource sets were considered for each case.

The obtained results are shown in Table 1. The table presents the characteristics of the best-case and worst-case DP. In addition, there are six more DPs for each of the five resource sets considered. Columns 2 and 5 present the latency results of a traditional list scheduling algorithm. Columns 3 and 6 offer the latency of the solutions produced by the proposed method. Columns 4 and 7 indicate the relative latency improvement of the method as compared to list scheduling. Column 8 presents the relative difference of the latencies for the worst and best case DPs. The last column indicates the execution time of the algorithm. Results show that the proposed method achieves on average a latency improvement of 29% as compared to list scheduling. Also, best-case latency differs on average by about 35% from the worst case. The execution time is reasonably large even for bigger MDGs.

ii. Adaptation policy design. We developed grids of 9, 18, and 27 sensor nodes (SN). Each SN had three modules as follows: one sampling module (Producer-S) with 5 different DPs, one input network module (Producer-N) with 3 DPs, and one output network module (Consumer) with 2 DPs. All DPs represent different latency - power trade-offs. The assumed Data Aggregation Pattern (DAP) included 8 different paths in the network. The optimized cost function included three terms, total power consumption, overall data loss, and size of the required buffers, with the last two being more important. Constraints were minimum data input rate and latency.

The experiments offered insight on how the problem goals and constraints influence the steady-state probabilities of different DPs, hence the nature of the adaptation process. We observed that optimal solutions included many DPs with a high power consumption - latency ratio. The data precision constraint is important in selecting the DPs of the sensing component ($Producer - S$). Tightening the power constraint increases the loss as the algorithm picks for the output network module the slower DPs but with lower power. The data

Ex.	Delay	Total Loss	Max. Loss	Total Power	Total Buffer	Time (sec)
Net 9	4.1	8.85	2.02	4.61	19.90	0.06
Net 9	5.0	7.97	2.18	4.60	3.56	0.06
Net 9	6.5	8.75	1.96	3.65	0	0.06
Net 18	4.4	22.74	5.64	12.14	30.54	0.26
Net 18	5.5	20.33	3.91	11.12	4.47	0.26
Net 18	8.0	18.66	3.91	10.99	1.48	0.26
Net 27	5.6	35.62	3.35	16.94	27.63	0.46
Net 27	6.5	31.50	3.91	15.33	6.42	0.46
Net 27	9.0	29.08	3.91	14.19	3.95	0.46

TABLE II
ADAPTATION POLICY CHARACTERISTICS

Ex.	Loss+Pow. Opt.			Loss Opt.		Pow.
	Delay	Loss	Power (μW)	Loss	Power (μW)	Sav. (%)
Net 27	84k	24	29010	13	42021	30
Net 27	86k	24	28491	10	42494	32
Net 27	90k	24	27452	8	43046	36

TABLE III
PSOC-BASED SENSOR NETWORK ADAPTATION

loss constraint has an important influence on the steady-state probabilities of the output network states. Also, increasing the output rate of nodes does not necessarily reduce data loss. Higher output rates might over-constrain the entire path, which can increase the loss of subsequent nodes, and thus lead to higher overall loss and power consumption.

Table 2 offers quantitative insight into the experiments. The second column indicates the latency (delay) constraint set for each example. The third column shows the total data loss in the network. The fourth column presents the highest loss at a single node. The fifth column indicates the total power consumption, and the sixth column the total buffer space of the SNs. The last column presents the execution time of solving the linear optimization problem by solver lp_solve. The results show that the methods scale well with the size of the networks. The total loss and required buffer space are also relatively low. Specifically, between 14% and 33% of the SNs did not experience any loss. For networks with 27 nodes, between 22% and 50% of the SNs did not require local buffer space to accommodate the rates of their producers and consumers. Power consumption decreases as the latency constraint is relaxed, but the decrease is small due to the smaller weight of power consumption in the cost function.

iii. PSoC based grid sensor network. The grid sensor network included 27 nodes based on the PSoC embedded processor [13]. Each node sensed temperature using a temperature sensor, and then communicated the value to its neighbors defined by the eight, considered paths. The sensing module, input network module, and output network module had 3 DPs each. For each DP, we measured the data rate, the number of clock cycles needed to execute the C code of the DP, and the power consumption.

The adaptation policies were produced for two optimization situations: (i) optimizing both data loss and power consumption, and (ii) optimizing only the data loss. Both cases considered constraints on the minimum rate of sampling temperature and the latency of sending the sampled data to the target node. Table 3 shows the obtained results. In the first case, reducing the latency constraint does not change the overall data loss as the adaptation policies tend to use only slow DPs with low power consumption. This is also justified by the fact that temperature sampling is slow, up to 64963 clock cycles. The second case optimizes the overall data loss but at the penalty of increasing power consumption, as shown in column seven. Power consumption is increased by as much as 36% as compared to the first case.

V. CONCLUSION

This paper presents a systematic methodology for designing the adaptation policies of reconfigurable sensor networks. The design flow includes two main steps: generation of alternative DPs representing different performance-cost trade-offs, and finding the switching rates between DPs to achieve effective adaptation. DPs are found using an exploration algorithm that performs combined mapping and scheduling. Adaptation policies are found by solving a Linear Programming problem describing adaptation as a Continuous-Time Markov Process. Experiments show that the methods scale well with the size of the networks. The produced policies offer small data loss, and have low buffer memory requirements. Finally, the policies can reduce power consumption on average by about 36% as compared to policies that are not power aware.

REFERENCES

[1] C. Chong and et al., "Sensor networks: Evolution, opportunities, and challenges," *Proc. IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[2] K. Gummadi and et al., "Macro-programming wireless sensor networks using Kairos," in *Proc. International Conference on Distributed Computing in Sensor Systems*, 2005.

[3] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, January 2002.

[4] D. Bertozzi and et al., "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, 2005.

[5] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. DAC*, 2001.

[6] A. Hansson and et al., "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proc. CODES+ISSS*, 2005.

[7] G. Ascia, V. Catania, and M. Palesi, "Multi-objective mapping for mesh-based NoC architectures," in *Proc. CODES+ISSS*, 2004, pp. 182–187.

[8] S. Murali and et al., "A methodology for mapping multiple use-cases onto networks on chips," in *Proc. DATE*, 2006, pp. 118–123.

[9] N. Thepayasuwan and A. Doboli, "Layout conscious approach and bus architecture synthesis for hardware-software co-design of systems on chip optimized for speed," *IEEE Trans. VLSI*, vol. 13, no. 5, 2005.

[10] Z. Guz and et al., "Efficient link capacity and QoS design for network-on-chip," in *Proc. DATE*, 2006, pp. 9–14.

[11] S. Kallakuri and A. Doboli, "Customization of arbitration policies and buffer space distribution using Continuous Time Markov Decision Processes," *IEEE Transaction on VLSI*, vol. 15, no. 2, 2007.

[12] W. Meng and et al., "Towards a model and specification for visual programming of massively distributed embedded systems," in *Proc. of IEEE International Workshop on Robotic and Sensors Environments (ROSE)*, 2008.

[13] Cypress Semiconductor Corporation, "PSoC mixed signal array," *Document No. PSoC TRM 1.21*, 2005.