Exploiting Partially Reconfigurable FPGAs for Situation-Based Reconfiguration in Wireless Sensor Networks

Rafael Garcia, Ann Gordon-Ross, and Alan D. George
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL
{garcia, ann, george}@chrec.org

***Abstract*—Wireless sensor networks (WSNs) are typically composed of very small, battery-operated devices (sensor nodes) containing simple microprocessors with few computational resources. However, the rapidly increasing popularity of WSNs has placed increased computational demands upon these systems, due to increasingly complex operating environments and enhanced data-sensing technology. Whereas introducing more powerful microprocessors into sensor nodes addresses these demands, sensor nodes do not contain sufficient energy reserves to support these microprocessors. In this paper, we present a partially reconfigurable FPGA-based architecture and methodology to provide increased WSN flexibility and computational resources, resulting in superior power consumption and performance compared to a microprocessor capable of satisfying similar demands.**

Keywords - Kalman filter; virtual architecture; reconfigurable computing; FPGA; partial reconfiguration

I. Introduction and Motivating Example

Wireless sensor networks (WSNs) are composed of a sensing array (multiple sensor nodes) distributed across an area with the purpose of observation and data collection. Each sensor node consists of a microprocessor coupled with wireless communication technology and a sensor, both of whose ranges determine a node's communication and sensing boundaries. Within these boundaries, a sensor node may be responsible for detecting, tracking, and/or monitoring single or multiple environmental points-of-interest (*targets*) using specialized algorithms or applications (*modules*).

Growing demands for more efficient WSNs have forced WSN designers to explore onboard processing solutions [2]. Onboard processing reduces wireless communication bandwidth, and thus power consumed due to this communication, by performing data processing on the sensor node, and transmitting only the results. One option, field-programmable gate arrays (FPGAs), have become the focus of recent research [6]. FPGAs offer increased performance compared to microprocessors and increased flexibility compared to ASICs [7], while maintaining low power consumption.

However, an FPGA-based system that considers multiple modules is more difficult to design than a microprocessor-based system. Since WSN environments are dynamic in nature, the quantity and type of targets may change frequently, requiring many system reconfigurations – *situation-based reconfiguration*. Thus, *modular systems* are required to provide flexibility with respect to executing module quantity and type. However, even though FPGAs are reconfigurable in nature, such modular dynamics are not inherent, since reconfiguration typically does not occur during execution. Many other challenges exist when taking full advantage of parallelism in modular FPGA-based sensor nodes.

Considering an environment's dynamic nature, the primary challenge lies in choosing how many modules and which types of modules to include on a sensor node. One potential solution creates many predefined module mixes for different situations and a bitstream (i.e., FPGA configuration file) for each situation. Choosing an appropriate module mix is non-trivial, as the number of combinations is exponential. Additionally, FPGA runtime reconfiguration between different module mixes interrupts execution. Fortunately, partial reconfiguration (PR) in FPGAs addresses many of these limitations [4]. PR enables selective region reconfiguration without system disruption. This module isolation reduces bitstream storage and communication requirements, since only the data associated with the particular PRR is required.

In this paper, we propose a methodology and modular architectural framework for situation-based reconfiguration in WSNs using PR-capable FPGAs. Using a Kalman filter as the base computational kernel, results show clear benefits for a PR-based FPGA implementation over a microprocessor or static FPGA system implementations. These benefits include power consumption reductions, consistent performance when tracking multiple targets, and significant cost savings.

II. Related Work

Both WSNs and FPGAs have been popular disjoint research topics, but their combination is an emerging field [7], and thus there exists little research. New and innovative WSN applications demand longer battery life, faster responses, and enhanced results. FPGAs, and in particular PR-capable FPGAs, provide a logical next step to satisfy these demands by providing a great balance of performance, parallelism, cost, and power.

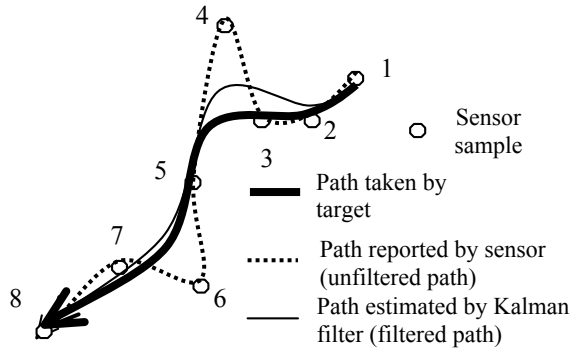

Figure 1. Example depicting how Kalman filtering improves path prediction by removing noise (sensor samples 4 and 6).

A. Wireless Sensor Networks

The first step towards situation-based FPGA reconfiguration is a state-of-the-art method to support situation-based reconfiguration via microprocessors. Agilla [3] is an architecture and methodology enabling WSN users to create special programs called *mobile agents* (a.k.a. modules) that wirelessly migrate across the sensor node network, performing application-specific tasks. Each sensor node consists of a MICA2 mote with an 8MHz Atmel 128 microprocessor and 4KB of data memory. Even though the goal of their test system is wildfire tracking, Agilla is applicable to other WSNs. However, since Agilla microprocessors are low power and have limited processing capabilities, Agilla is unable to process advanced sensor data, such as video feeds, which are quite common in target tracking and identification.

FPGA implementations of Kalman filters (a popular target tracking kernel) demonstrate excellent performance [5]. In [1], a system that combined feature selection and Kalman filtering in an FPGA completed the tracking process for hundreds of features within 2-12 milliseconds, resulting in an average rate of 20 microseconds per feature. However, these implementations did not consider targets with different characteristics, such as widely varying speeds and tracking criticality, which would significantly increase processing time.

B. FPGA Partial Reconfiguration

PR provides a method to reconfigure selected regions of the FPGA fabric while the remainder of the device remains active. The FPGA fabric is partitioned into two or more partially reconfigurable regions (PRRs) (that can load arbitrary modules) plus one static region that does not change. In contrast to full FPGA device configuration, which requires *full bitstreams* (configuration data for the entire device), PR-capable FPGAs use *partial bitstreams*, only specifying the configuration data for a particular PRR.

The most significant PR benefit is PRR reconfiguration without halting execution of the entire device. This isolated reconfiguration is beneficial when critical system tasks such as communication links, timers, managers, etc. must remain operational at all times. In the case of WSNs, active modules could be tracking critical targets and should not be halted while loading new modules.

TABLE I: SITUATION-BASED KALMAN FILTER VARIATIONS

Module type	Use conditions	Deviation from base module
Base Kalman	Generic	None
Low power	Standby, slow targets	Clock frequency, sample rate
High power	Fast moving, high priority targets	Clock frequency, sample rate
Airborne-target module	Airborne target tracking	Stereo camera for 3D coordinates
Noisy-target module	High-noise readings: defective sensor, rainy day, night tracking	Fixed Kalman gain

III. TARGET TRACKING AND KALMAN FILTERS

Target tracking is a path/trajectory prediction method for environmental targets such as vehicles, missiles, animals, hostiles, or even unidentified objects. In order to make this prediction, sensors measure and record target characteristics such as speed, past trajectory, acceleration, etc. However, the prediction accuracy is only as reliable as the measured data and this data is frequently noisy. Fortunately, Kalman filtering provides a method to reduce sensor measurement noise, and thus provides a more accurate prediction of a target's future path. In this section, we will give a brief example and background on Kalman filtering, discuss variations in Kalman filtering, and PR-capable, FPGA-specific filtering benefits.

A. Example and Background

A Kalman filter's main purpose is to estimate dynamic system state in a noisy environment. Specifically, we will examine a dynamic system consisting of a moving target. Figure 1 shows an example of how noise can affect a target's predicted path. The solid bold line represents the target's actual path. The numbered circles represent sequential sensor samples representing the locations where the sensor believes the target is. Samples 4 and 6 are noisy (incorrect) samples. Using these noisy samples, interpolation results in the dashed line (unfiltered path) as the target's predicted path. However, using a Kalman filter removes the noisy samples and results in the thin solid line (filtered path) as the improved predicted path.

The Kalman filter can work independently of the other sensor nodes if required, although additional information from neighboring sensor nodes produces enhanced results. Since WSNs can be unreliable at times, due to insufficient power reserves or harsh environmental conditions, this inter-node cooperation is a favorable aspect.

B. Situation-Based Kalman Filter Variations

Table I shows a small subset of situation-based Kalman filter variations (*module type*), conditions where each module would be most effective (*use conditions*), and how each module deviates from the base Kalman filter module (*deviation from base module*). For some Kalman filters, computational terms such as the Kalman gain and matrix covariance can be pre-computed for certain target types, further increasing both module variation and performance.

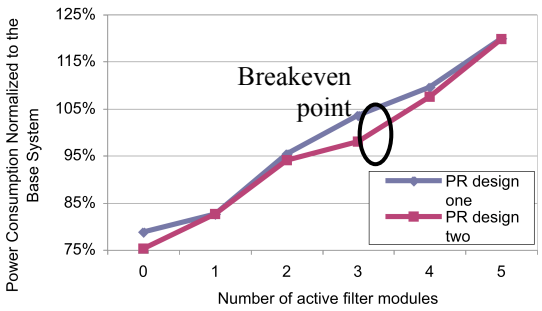

Figure 2: Power consumption normalized to the base system for PR designs one and two for a varying number of modules.

IV. Architectural Framework

In order to establish communication between arbitrary module PRRs as the system configuration changes, a flexible module communication architecture is required to facilitate dynamic module loading and unloading. We developed a virtual architecture that facilitates inter-module communication, regardless of module location, size, or clock domain.

A. Virtual Architecture

VAPRES (pronounced "vapors", the Virtual Architecture for Partially Reconfigurable Embedded Systems) is a virtual architecture developed for PR-capable FPGAs to provide a flexible and dynamic module communication layer. Each PRR represents a different clock domain, such that each PRR can be independently clocked. The VAPRES system control region includes a soft processor to serve as the central controlling agent, a flash controller core to read and store module partial bitstreams, and numerous peripherals for external device communication. In addition, real-world WSNs would include wireless communication controllers, battery monitoring cores, and digital clock managers for multiple clock domain management. Since these components would not change at runtime, they reside in the FPGA's static region along with any other non-reconfigurable critical system components.

VAPRES and the underlying communication architecture provide an essential enabling methodology for dynamic, situation-based PR. Whereas our current VAPRES architecture is not PR-specific, we are currently developing the PR-based enhancements.

B. Situation-Based Reconfiguration Methodology - Dynamic Module Loading and Unloading

The VAPRES central controlling agent orchestrates module loading and unloading, which involve making runtime decisions on when to place a module inside the PRRs (online scheduling) and which specific PRR implementation to use (online placement). Inter-module communication is necessary when a module optimized for noisy measurements has determined the correct target type and wants to transmit information to a new module that can better track the target. VAPRES is able to seamlessly handle this inter-module communication.

While the sensor node is not actively tracking a target, a target detection module is responsible for new target detection. When a new target is detected, the central controlling agent loads an appropriate module and initiates necessary communication establishment based upon three possible scenarios:

1. A basic tracking module processes the sensor input for new targets. New targets are evaluated and the appropriate specialized module is loaded. The basic module then returns to standby mode and waits for a new target.

2. The central controlling agent loads all potential tracking modules in succession into a single PRR, and each module tracks the target for a short evaluation period. The module that best tracked the new target is reloaded and responsible for tracking that target.

3. The controlling agent loads all possible modules in parallel. After a short evaluation time, the module that best tracks the target would remain loaded.

Scenarios 2 and 3 rely on loading tracking modules for module evaluation before determining the most appropriate module. Scenario 1 performs target evaluation without loading evaluation modules, eliminating module evaluation overhead. Since each of these scenarios has tradeoffs in terms of power consumption, latency, tracking accuracy, and resource usage, VAPRES enables designers to make these tradeoff decisions based on system goals.

V. Results

In this section, we evaluate the VAPRES architecture and situation-based reconfiguration using three metrics: processing performance, power consumption, and resource utilization. Even though we implemented the Kalman filter on a high-end FPGA, which might not be used in a real-world application, our results are equally applicable to low-end FPGAs and provide motivation for power, performance, and resource utilization improvements.

A. Experimental Setup

Our experimental setup used a Xilinx Virtex-4 FX100 FPGA. Although Virtex-5 FPGAs offer better performance and power, we used a Virtex-4 due to its PR design flow maturity. We measured power consumption using the Xilinx XPower analysis tool. We assumed a 12% average signal activity. Since typical design activity ranges from 6-12%, our power estimates are pessimistic.

For our WSN system, we modeled a situation with a maximum of five simultaneously tracked targets, thus the system contained five PRRs for filter modules. Our system design contained only the VAPRES architecture and Kalman filters. Image processing hardware, sensor interface, and communication interface requirements were assumed to be necessary components regardless of the system configuration, and are therefore not included in our measurements.

In order to provide FPGA resource requirements for a variety of system situations, we constructed two PR design test cases. *PR design one* (PR-D1) used dedicated FPGA multiplier units, and *PR design two* (PR-D2) implemented multiplication using logic. The *base system* consisted of five copies of *design one* without the VAPRES architecture.

TABLE II: RESOURCE UTILIZATION FOR PR DESIGN ONE (PR-D1) AND PR DESIGN TWO (PR-D2) ON VIRTEX-4 FX100

Resource type	Amount used		% of device	
	PR-D1	PR-D2	PR-D1	PR-D2
Slice Flip Flops	5566	5566	6.6%	6.6%
4-input LUTs	6623	19193	7.9%	22.8%
DSP48 multipliers	70	0	43.8%	0%
Block Ram	20	20	5.3%	5.3%

B. *Processing Performance, Power Consumption, and Resource Utilization*

Since the VAPRES system enables multiple clock domains, we evaluate the maximum attainable clock frequency for the architecture independently from the PRRs. Since all PRRs have similar layouts (equal number of allocated resources), all PRRs have the same maximum attainable clock frequency, which is independent of the filter modules.

The VAPRES maximum attainable clock frequency is 209 MHz, while the Kalman filter modules reach 87 MHz and 68 MHz for PR-D1 and PR-D2, respectively. Since processing one input sample takes two clock cycles, the maximum clock frequency represents half the maximum sample processing rate. Whereas Agilla's [3] sample processing times reached 6-7 seconds, our sample processing times range from 23-30 nanoseconds.

Figure 2 shows power consumption of the two PR designs normalized to the base system for a varying number of active filter modules. The base system power consumption was 1.794 Watts. The figure reveals the breakeven point at three active modules. The breakeven point is the point where the PR design and the base system have equal power consumption. PR designs with fewer active modules consume less power than the base system, while PR designs with more active modules consume more power than the base system. Figure 2 also reveals that multiplier unit availability has essentially no impact on power consumption. It is worth noting that the results from Figure 2 only compare the differences in dynamic power consumption, since static power across all cases, regardless of PR, remains the same. The clock to PRRs is simply disconnected when not in use, eliminating only dynamic power consumption (and not static power).

These results are somewhat expected. PR provides enhanced design flexibility at potentially less power and resource requirements. Since the base system and the PR designs all contain five filter modules, at full capacity, the PR designs essentially provide equal functionality as the base system with additional PR overhead. However, the benefit of PR for WSNs is that the system will likely not be in high-power situations very often. Simply because a node provides the ability to track five targets simultaneously does not mean that the node will continuously track five targets. Thus, PR provides both reduced average power consumption and increased battery life when the system is not at peak operating capacity. Quantifying these savings is very difficult as they are entirely dependent on actual environmental situations such as target frequency, criticality, and complexity.

Table II shows the number of device resources used (amount used) and the percentage utilization of these resources (% of device) on the Virtex-4 FX100 for both PR designs. These numbers provide an easy method in which to evaluate alternative FPGA devices to determine if a candidate device has sufficient resources for a five-module VAPRES system. We point out that these numbers do not include resources required for processing hardware, sensor interfaces, and communication interfaces. Table II shows that the required resources for PR-D2 is nearly one half of those required for PR-D1 due to the replacement of dedicated multiplier units with logic. Despite the increase in LUT usage for PR-D2, the elimination of dedicated multiplier units provides a 45.4% reduction in device usage with a 21.8% performance penalty for this tradeoff.

VI. CONCLUSIONS AND FUTURE WORK

Future WSNs will continue to demand increasingly greater performance and lower power consumption. FPGAs are prime candidates to fulfill these requirements. We have presented a design methodology with architectural support for exploiting PR-capable FPGAs in situation-based reconfiguration for WSNs with transparent dynamic module allocation and inter-module communication establishment. VAPRES provides a critical stepping-stone for creating powerful, flexible WSN designs capable of satisfying increasing computational demands. Results showed that our situation-based reconfigurable methodology provides fast sample processing rates in the MHz range with a 5-25% reduction in power consumption compared to the fixed five-module design with two or fewer active targets. Furthermore, the low device utilization of our methodology makes our methodology highly amenable to small devices.

VII. ACKNOWLEDGMENTS

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC-0642422. We gratefully acknowledge tools provided by Xilinx.

VIII. REFERENCES

[1] Bissacco, A., S. Ghiasi, M. Sarrafzadeh, J. Meltzer, and S. Soatto. "Fast visual feature selection and tracking in a hybrid reconfigurable architecture." *Proceedings of the Workshop on Applications of Computer Vision (ACV)*. 2006.

[2] Dreicer, Jared S., Anders M. Jorgensen, and Eric E. Dors. "Distributed Sensor Network With Collective Computation For Situational Awareness." *AIP,* 2002. 235.

[3] Fok, Chien-Liang, Gruia-Catalin Roman, and Chenyang Lu. "Mobile Agent Middleware for Sensor Networks: An Application Case Study." *Information Processing in Sensor Networks, 2005*, 382-387.

[4] Hymel, R., A. George, and H. Lam, "Evaluating Partial Reconfiguration for Embedded FPGA Applications," Proc. High-Performance Embedded Computing Workshop, MIT Lincoln Lab, Lexington, MA, Sep. 18-20, 2007

[5] Lee, C. R., and Z. Salcic. "A Fully-hardware-type Maximum-parallel Architecture for Kalman Tracking Filter in FPGAs." *International Conference on Communications and Signal Processing,* 1997. 1243-1247.

[6] Todman, T.J., G.A. Constantinides, S.J.E. Wilton, O. Mencer, W. Luk, and P.Y.K. Cheung. "Reconfigurable computing: architectures and design." *IEE Proceedings: Computer & Digital Techniques,* Vol 152, No. 2, March 2005. 193-207.

[7] Wemekamp, John. "Emerging Trends in Mil/Aerospace Embedded Systems." *Electronic Component News*, May 2007: 27-29