In Proceedings of the International Workshop on Distributed Sensor Systems, pages 95-1-95-6, Honolulu, Hawaii, August 2007.

Design Techniques for Streamlined Integration and Fault Tolerance in a Distributed Sensor System for Line-crossing Recognition

Chung-Ching Shen, Roni Kupershtok, Shuvra S. Bhattacharyya, and Neil Goldsman

Dept. of Electrical and Computer Engineering, and Institute for Advanced Computer Studies
University of Maryland
College Park, USA
{ccshen, akuper, ssb, neil}@eng.umd.edu

Abstract — **Distributed sensor system applications (e.g., wireless sensor networks) have been studied extensively in recent years. Such applications involve resource-limited embedded sensor nodes that communicate with each other through self-organizing protocols. Depending on application requirements, distributed sensor system design may include protocol and prototype implementation. Prototype implementation is especially useful in establishing and maintaining system functionality as the design is customized to satisfy size, energy, and cost constraints.**

In this paper, we present a streamlined, application-specific approach to incorporating fault tolerance into a TDMA-based distributed sensor system for line-crossing recognition. The objective of this approach is to prevent node failures from translating into failures in the overall system. Our approach is specialized and light-weight so that fault tolerance is achieved without significant degradation in energy efficiency.

We also present an asynchronous handshaking approach for providing synchronization between the transceiver and digital processing subsystem in sensor node. This provides a general method for achieving such synchronization with reduced hardware requirements and reduced energy consumption compared to conventional approaches, which rely on generic interface protocols.

We demonstrate the capabilities of our approaches to fault tolerance and transceiver-processor integration through experiments involving a complete prototype wireless sensor network test-bed, and a distributed line-crossing recognition application that runs on this test-bed.

I. Introduction and Related Work

Distributed sensor systems, such as wireless sensor networks, address a great diversity of sensing applications, including habitat monitoring, environment observation, and battlefield surveillance [5]. In a variety of important applications, wireless sensor nodes are densely deployed in areas that are dangerous or otherwise inaccessible to humans, and communicate with one another through self-organizing protocols. Often, when designing a distributed sensor system, the size of individual sensor nodes should be small enough so that they can be hidden from the environment easily. Properties of low energy and power consumption are also important due to the requirement of extended system lifetime.

Time division multiple access (TDMA) protocols are often applied in small scale wireless communication systems [2, 11] due to their simplicity and low power communication patterns. Furthermore, with TDMA, collision avoidance can be guaranteed throughout the system. Based on various application-specific scenarios, distinct fault tolerant capabilities for TDMA-based systems are considered [1, 3]. In this paper, we introduce a self-organizing fault tolerance approach for a distributed sensor system that performs line-crossing recognition. We demonstrate the fault tolerance feature of the proposed system and discuss how the approach can be generalized to similar TDMA-based systems.

In most research and development projects involving wireless sensor network implementation, off-the-shelf components have been used. In general, these components include microcontrollers, transceivers, and sensors (e.g., see [4, 6]). Our final goal that we are working towards is to base the sensor node platforms in our distributed sensor system on application-specific integrated circuits (ASICs). An intermediate stage that we have completed is a sensor node prototype that is based on a field programmable gate array (FPGA). We use the FPGA to implement the digital processing core that controls the sensor node. In addition, we use a commercial transceiver for wireless communication, and an off-the-shelf acoustic sensor for sensing tasks. This prototype system provides the complete functionality for a sensor node in a distributed sensor system application. The purpose of using the FPGA is to build and validate a prototype before customizing the digital subsystem for ASIC development. The ASIC will be based on a customized digital circuit for the targeted application of line crossing recognition, and therefore, size and power constraints can be significantly reduced compared to an implementation that is based on a general-purpose microprocessor or microcontroller.

When building a sensor node by combining various subsystem platforms, as described above, synchronization must be handled across the different platforms, and such synchronization requires special care when the platforms employ separate clocks. The conventional approach to this synchronization problem is that both platforms negotiate with each other via generic synchronization protocols, such as universal asynchronous receiver-transmitter (UART) or serial peripheral interface (SPI). To run these protocols, the platforms that are being interfaced require additional hardware requirements, which may increase size and energy usage for the prototype implementation.

Thus, in this paper we also introduce an asynchronous approach for interfacing two platforms for sensor node prototyping when the platforms have separate clocks. Unlike generic interfacing methods, such as UART and SPI, our approach is specialized to the specific needs of sensor node integration, and therefore involves less complexity, and therefore less hardware and energy cost. We demonstrate this approach by interfacing an FPGA-based digital processing platform to an off-the-shelf transceiver platform. We show in our experiments that through our interfacing approach, the FPGA and transceiver platforms interact asynchronously in a robust manner.

Various researchers (e.g., see [7, 8]) have used FPGAs as hardware platforms to implement sensor nodes; however, these works have not addressed the problem of streamlining the FPGA-transceiver interface.

In this paper, we not only address specific theoretical and practical aspects of a proposed fault tolerant distributed sensor system, but we also demonstrate a complete system prototype implementation that is based on off-the-shelf components, and customized digital processing functionality that is implemented on FPGAs.

II. Distributed Line-crossing Recognition System

Our proposed approach to fault tolerance is based on our developed distributed sensor system for line-crossing recognition (LCR) [9]. The system presented in [9] is the original version of our LCR prototype, without the features of FPGA-based digital system implementation, streamlined asynchronous integration, and fault tolerance that we describe in this paper. For example, our system can be applied by deploying the sensor nodes in a circle inside a room. Each sensor node platform is equipped with an acoustic sensor. Sensors sense continuously, and an analog-to-digital converter (ADC) is used to convert sensed signals to raw samples. Each raw sample is then compared to a pre-defined threshold that is tunable to fit various circumstances. We design a simple finite state machine to verify whether detected signals that are above the threshold present false detections. If the threshold is exceeded and the test result for false detection is negative, then the system determines that a sound source has been detected. The system then recognizes and provides information about when and where a subject has crossed the circle. All sensor nodes in the system periodically reach consensus in deciding whether or not a subject ("intruder") has crossed a specific boundary ("line") in a noisy environment that is being monitored. Furthermore, upon detecting an intrusion, the system determines where the line was crossed (i.e., between which nodes in the line).

We assume that all nodes in the system are deployed within radio range of each other and all node-to-node communications are based on a ring topology. Since such a system is fully distributed without using any base station, a self-organizing protocol is required to make sure that all sensor nodes communicate with each other effectively. We have designed an efficient, wireless TDMA protocol so that every node receives and transmits at designated time slots, and sleeps during other times for saving energy. All N nodes within the system run the distributed algorithm, and reach a consensus based on local decisions of C nodes while a subject is being detected ($C \leq N$). C is a pre-determined parameter that allows the designer to control a trade-off between recognition accuracy and communication requirements. For more details on the design and analysis of this distributed sensor system, we refer the reader to [9].

Our proposed distributed system uses a TDMA-based communication protocol that consists of two stages: synchronization and communication. Note that in the synchronization stage, each non-functioning node is initialized to become a functioning node and carry out initial clock synchronization with its neighbor node in this stage. The clock synchronization scheme involves having each node adjust its own clock to the clock of its neighbor node based on the time it receives a valid packet from its neighbor node as well as on the pre-defined TDMA time schedule. During the synchronization stage, nodes are synchronized with each other and know whether the whole system is synchronized or not. Whenever a node is powered-on, it is in the synchronization stage with periodic iterations of transmitting one packet and continuously trying to receive packets from its neighbor. Ideally each powered-on node transmits one packet and receives at most one packet from its neighbor in each iteration. Each node in the system has a unique identifier (id). Node i ($0 \leq i \leq N-1$) stays in the synchronization stage until it receives a packet from its previous node $(i-1)$ (i.e., node i is synchronized with node $i-1$) and knows that the whole system is synchronized (i.e., all nodes are synchronized with their neighbors). Afterwards, node i enters the communication stage, where it periodically communicates with other nodes in the network.

During the communication stage and based on a pre-defined TDMA time schedule, every node receives, computes, transmits, and stays in an idle mode periodically. Moreover, every time a node receives a valid packet from its neighbor, it carries out the clock synchronization scheme with its neighbor so that clock drift is prevented. Figure 1 presents an example of a TDMA time schedule for the distributed system with four nodes. On the left, Figure 1 illustrates the synchronization stage, where each node can be turned on at an arbitrary time.

According to our design specification, all sensor nodes in the system follow the same pre-defined TDMA-based communication pattern to satisfy the system functionality for line-crossing recognition. However, without considering fault tolerance, any node failure (e.g., a node stops processing and communicating due to lack of energy) during the communication stage in general causes abnormal termination of all other nodes in the system. That is, the system fails whenever there is a node failure scenario. Thus, in order to protect against system failure, we have developed and integrated a distributed fault tolerance protocol into our distributed sensor system. Using our fault tolerance approach, system functionality is maintained when arbitrary (proper) subsets of the network nodes fail. Thus, each node failure leads in general to a decrease in system accuracy, as opposed to a failure in overall system operation.

III. TDMA-based Fault Tolerance Approach

Our overall approach to fault tolerance is divided into two parts. The first part is used in the synchronization stage, and the second part runs for system failure prevention in the communication stage.

A. Synchronization Stage

The approach that we use in the synchronization stage is to have all of the nodes separately determine whether or not the system is synchronized, in addition to the initial node clock synchronization process discussed previously. The system is synchronized only if all of the functioning nodes are synchronized and agree on this situation. Once the system is synchronized, the algorithm of system failure prevention at the communication stage can be activated.

During the synchronization stage, there are only functioning (powered-on) nodes and powered-off nodes. Moreover, powered-on nodes will be synchronized and will enter the communication stage in a relatively short period of time. Our algorithm for system failure prevention is not incorporated in the synchronization stage. This is because a node might consider a powered-off neighbor as a failed node. Whenever a node fails in the communication stage, the distributed system will be reorganized automatically within the time period of single TDMA frame.

The fault tolerance algorithm that we employ in the synchronization stage operates in the following way. Suppose that there are N nodes in the system, and every node has a unique identifier (id) i such that $0 \le i \le N-1$. Initially, once node i is turned on, it periodically transmits a packet to node $((i+1) \bmod N)$, every s seconds. Such a packet includes a control message field, an id field, and a global counter field ($globalC$) with an initial value of 1. At synchronization stage,

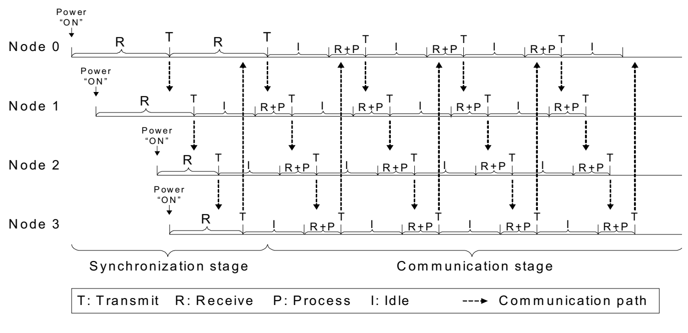


Figure 1. TDMA-based communication pattern for a four node example of the proposed LCR system.

$globalC$ counts the number of nodes that are on and are synchronized. In addition, node i keeps an internal synchronization flag (F_{sync}), which is set to zero when the node is powered-on. F_{sync} indicates that node i knows whether all the nodes in the system are synchronized. Note that during the synchronization stage, some nodes might be on while others might be off. Therefore, the overall distributed system is not necessarily synchronized at this time.

While node i transmits a packet every s seconds, where s denotes the duration of TDMA time frame, it continuously tries to receive a packet from node $((N+i-1) \bmod N)$ (i.e., from its "left" neighbor in the circular, virtual linkage of nodes based on their identifiers). Whenever node i receives a packet from node $((N+i-1) \bmod N)$ with $globalC < (N-1)$, it reads the associated $globalC$ value, increments this value by 1, and transmits it within a data packet to node $((i+1) \bmod N)$.

Whenever a node j ($0 \le j \le N-1$) is the first to receive a packet from its predecessor (i.e., from node $((N+j-1) \bmod N)$) with $globalC = N-1$, node j sets its internal F_{sync} to 1. Then node j transmits a packet to node $(j+1) \bmod N$ with a unique control message α, and $globalC$ is set to 0. That is, at this point, node j knows that all the nodes are on and that they are synchronized with their neighbors. Therefore, node j starts the process of informing all other nodes in the system that all the nodes are on and are synchronized. It does this by transmitting the control message α to its neighbor.

Now suppose that a node k ($0 \le k \le N-1$ and $k \ne j$) whose F_{sync} value is 0 receives a packet from node $((N+k-1) \bmod N)$ with the control message α. Then node k sets its F_{sync} value to 1, and transmits a packet to node $((k+1) \bmod N)$ with the control message α, and with a $globalC$ value of 0.

When node j (i.e., the first node that set its F_{sync} value to 1) receives a packet from node $((N+j-1) \bmod N)$ with the control message α, then node j knows that all the nodes in the system have received α. Then, node j starts the communication stage by transmitting a packet to node $((j+1) \bmod N)$ with the control message β and with a $globalC$ value of zero. At this point, all the nodes are synchronized in the system.

Note that α is transmitted as long as there are still nodes that are not synchronized, and β is transmitted once all the nodes are on and are synchronized. Moreover, the time period over which the whole system remains in the synchronization stage must be larger than the time difference between when the first and the last nodes join the system plus an additional s seconds. For further details of control message usage in the communication stage, we refer the reader to [9].

B. Communication Stage

Our targeted distributed system is like a fault tolerant data packet passing system with all nodes equipped with our proposed fault tolerance protocol. At any moment, only one node is transmitting a data packet and only one node is receiving a data packet due to the imposed TDMA pattern. Both the transmitting and receiving nodes are neighbors in the virtual, circular linkage nodes based on their unique identifiers.

If our TDMA-based distributed system consists of N nodes, the TDMA time frame of s seconds is divided into N slots, and each time slot lasts for a period of s/N seconds. During slot i ($0 \le i \le N-1$), node i transmits a packet, and node $((i+1) \bmod N)$ receives that packet in its receiving window. Here, the receiving window is defined as the longest time period allowed for receiving packets within a given TDMA time slot.

When node i does not receive packets from node $((N+i-1) \bmod N)$ (i.e., if node $((N+i-1) \bmod N)$ has failed), then node i starts a process to find a new neighbor. Based on the pre-defined TDMA schedule, node i shifts its receiving window from slot $((N+i-1) \bmod N)$ to slot $((N+i-2) \bmod N)$ and tries to receive a packet from node $((N+i-2) \bmod N)$ during the next TDMA time frame. If it succeeds, then node $((N+i-2) \bmod N)$ becomes the new neighbor of node i. Otherwise, node i shifts its receiving window, and tries to receive packets from nodes $((N+i-3) \bmod N)$, $((N+i-4) \bmod N)$, and so on in the succeeding TDMA time frames until it meets with success. Note that node $((N+i-1) \bmod N)$ is considered from this point onward as a non-functioning (failed) node.

When node i succeeds in receiving a packet from some node $((N+i-\delta) \bmod N)$ ($1 \le \delta \le N-2$), then δ becomes the new neighbor of i in the new, fault-adapted, virtual linkage structure of the remaining $(N-\delta)$ functional sensor nodes. All the nodes between nodes i and $((N+i-\delta) \bmod N)$ (non-inclusive) are considered from this point onward as being non-functioning nodes.

In Section V, we will present a concrete node failure scenario to illustrate our approach to fault tolerance.

IV. Proposed Asynchronous Prototyping Approach

Without generic synchronization protocols, such as UART and SPI, synchronous interfacing of separately-clocked platforms in a sensor node prototype system requires a master clock that is potentially much faster than other clocks in the system. Such a synchronous interfacing approach may cause major problems due to clock skew.

In this paper, we introduce a light-weight handshaking scheme that is appropriate for prototyping sensor node systems that involve separate clock platforms. This scheme avoids the aforementioned problems associated with synchronous interfacing, and it is more cost- and energy-efficient compared to conventional, generic approaches to asynchronous interfacing within sensor nodes.

Our asynchronous interfacing approach can be generalized to any platform pair that conforms to a master-slave structure. In our prototype implementation, the master platform is the field programmable gate array (FPGA) platform, and the slave platform is the transceiver platform. Since the FPGA and transceiver platforms execute tasks based on separate clocks, our proposed handshaking approach allows these two devices to interact asynchronously in a robust manner.

In our design, we use separate channels for the data, request, and acknowledgement signals. The FPGA platform runs as the main processor to deal with computation and control tasks, and the transceiver platform is controlled by the FPGA to execute communication tasks (i.e., to transmit and receive signals through the wireless channel). Therefore, in accordance with the TDMA-based protocol described in the previous section, the FPGA determines and sets control signals (e.g., requests and acknowledgments) to the transceiver for executing tasks in transmit (TX) and receive (RX) modes, or for changing its status to the idle mode.

Without loss of generality, the scenario of the proposed asynchronous approach between two separate clock platforms in TX mode is as follows. Whenever the FPGA is running in the TX mode and is ready to transmit, it sends a TX signal (W-TX) to the transceiver platform. The transceiver platform receives the W-TX (R-TX) signal, enters TX mode, acknowledges back (W-Ack) to the FPGA platform, and then monitors the request channel from the FPGA. Once the FPGA receives W-Ack (i.e., notification that the transceiver platform is ready in TX mode), it places a data bit on the data channel and changes its request signal (C-Req). Whenever the transceiver platform detects the C-Req signal, it reads a data bit from the data channel (R-Data), and then sends an acknowledgement back to the FPGA (W-Ack). Every time the FPGA receives a W-Ack in this way, it can place another data bit on the data channel and repeat the handshaking process described above until it successfully passes all the data bits that it needs to send at a given point in time. Once the FPGA passes all the data bits, the transceiver platform may enclose the data bits in appropriate packets and transmit the resulting packets through the wireless channel. Figure 2 illustrates the state transition graph for the handshaking schemes associated with control signals in TX and RX mode.

In the next section, which presents a case study of the design techniques presented in this paper, we demonstrate an implementation of our proposed handshaking approach in our prototype wireless sensor network for line-crossing recognition.

V. Experiments

As mentioned in Section III, all nodes in our line crossing recognition system are using the proposed fault tolerance algorithm when they are in the synchronization and communication stages. The run-time fault tolerance scenario in the communication stage is illustrated in Figure 3, where the system is illustrated using a 4-node example.

In Figure 3, the four nodes have unique identities 0, 1, 2, and 3; "TX" denotes transmitting a packet to the next neighbor node; and "RX" denotes receiving a packet from the previous neighbor node.

Figure 3(a) presents the normal communication pattern during the communication stage without any node failure. In Figure 3(b), node 0 fails. Therefore, node 1 (the neighbor of node 0) does not receive a packet from node 0 at the desired time slot. As a result, node 1 detects that node 0 has failed. In the next time frame, node 1 shifts its receiving window from slot 0 to slot 3 (See Figure 3(c)). Thereafter, node 1 receives packets from node 3, and node 0 is not considered functioning node anymore. There is no effect on the overall TDMA time frame

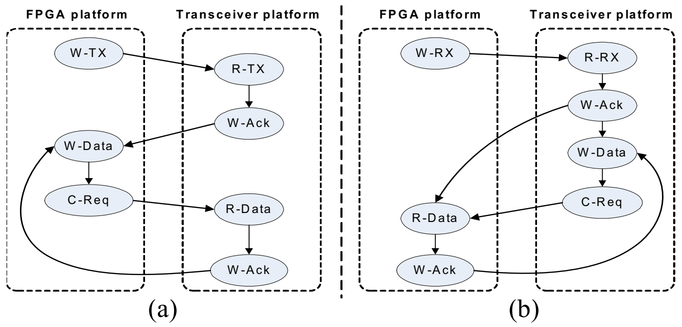


Figure 2. The state transition graphs for handshaking in (a) TX mode and (b) RX mode.

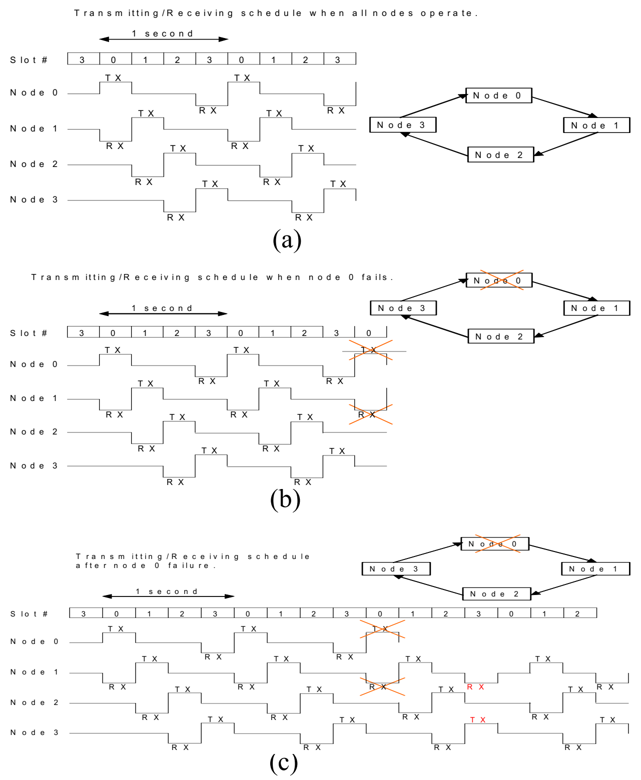

Transmitting/Receiving schedule when all nodes operate.
1 second
Slot #
Node 0
Node 1
Node 2
Node 3
TX
RX
(a)
Transmitting/Receiving schedule when node 0 fails.
(b)
Transmitting/Receiving schedule after node 0 failure.
(c)

Figure 3. A node failure scenario in our proposed fault tolerance distributed system.

duration s since the time slot for the failed node (node 0) simply becomes an unused slot in the revised TDMA schedule. That is, the TDMA time schedule is pre-defined initially, and is adapted as execution evolves based on changes in system status.

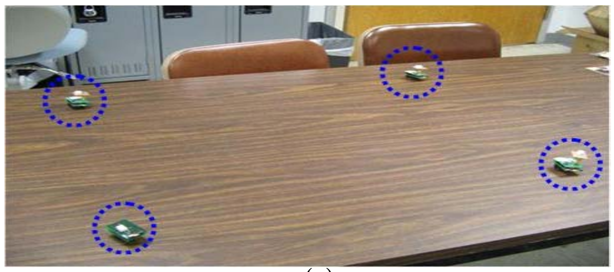
A. Experimental Setup using Software-based Control Implementation

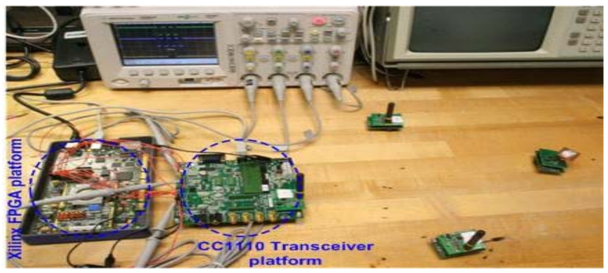
We have implemented and deployed our first system prototype for distributed line-crossing recognition application using off-the-shelf components, and embedded software implementation for the control and associated data processing functionality (i.e., for the complete digital subsystem functionality) within each sensor node.

Figure 4(a) shows the experimental deployment of the our first system prototype. This experimental setup includes 4 sensor nodes. Each sensor node is equipped with a Texas Instruments/Chipcon CC1110 (a single integrated circuit that contains a 16-bit RISC microcontroller and a 916MHz transceiver), a customized 916MHz antenna [10], and an acoustic sensor.

For setting up the TDMA-based protocol, one round of transmitting and receiving 4 packets within the system requires 1 second, and thus $s = 1\ second$. Therefore, the duration of each TDMA time slot is 250ms.

In our experimental system, every packet is received in the middle of its receiving window. This is to prevent collisions between the packet that was received in the previous slot and the packet that will be received in the next slot. For example, in our experimental setup with 4 nodes, 1 second time frame, and 250 ms time slot, the length of a receiving window is 250ms. Thus, every packet is received roughly 125ms after the beginning of its receiving window.


(a)



(b)

Figure 4. (a) First system prototype. (b) Second system prototype with an FPGA setup.

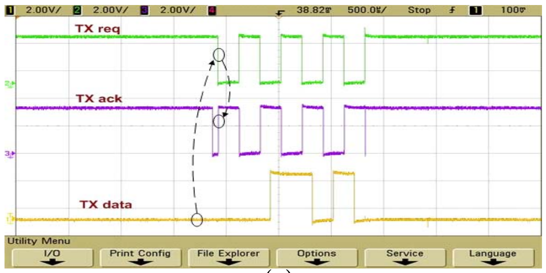
B. Experimental Setup using FPGA-based Control Implementation

We have also implemented and deployed a second system prototype for our distributed line-crossing recognition application using an FPGA platform for control functionality (See Figure 4(b)).

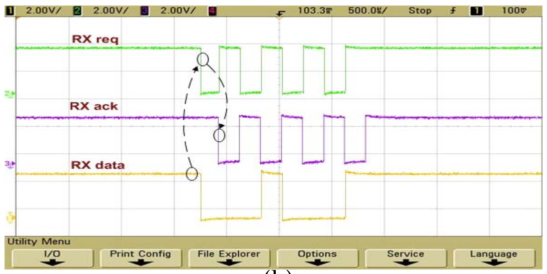
In this development, we use the Xilinx Virtex-4 FPGA as the control core for a sensor node. That is, the embedded software targeted to the microcontroller in the first prototype is replaced by the FPGA. Therefore, in the second prototype, we only use the transceiver part of the CC1110 device. This prototype implementation demonstrates a complete sensor node in our distributed sensor application with custom logic used to implement all control functionality. We have verified its correct operation (including both communication and computation) in conjunction with other nodes in the system.

All control and associated data processing for the application has been modeled and implemented in Verilog, the Verilog implementation has been synthesized onto the targeted FPGA device to demonstrate the sensor node behavior. For example, if the sensor node is ready to transmit data, the FPGA will first set up the data and then send a request signal to the transceiver. Conversely, each time the transceiver receives a request along with the associated data from the FPGA, it will send an acknowledgement back to FPGA to complete each transaction.

To interact between the FPGA and the transceiver for a sensor node using our proposed asynchronous approach, the handshaking protocol mentioned in Section IV is modeled and implemented in Verilog for the FPGA platform, and in C for the CC1110 microcontroller subsystem. Figure 5 shows snapshots captured from an Agilent DSO 6041A oscilloscope that depict handshaking interactions between the FPGA device and the CC1110 transceiver subsystem in either transmit (TX) and receive (RX) mode.



(a)



(b)

Figure 5. Handshaking interactions between FPGA and CC1110 platforms in (a) TX and (b) RX mode.

VI. Conclusion

In this paper, we have presented a light-weight fault tolerance approach for a TDMA-based distributed sensor system for line-crossing recognition. This approach prevents failures in individual nodes from translating into failures of the overall distributed sensor system. We have also presented an asynchronous handshaking approach for sensor node prototypes that involve separately-clocked platforms (e.g., to integrate different platforms that host digital processing and transceiver circuitry). Such integration allows designers to efficiently and robustly mix and match different platforms during prototype-level experimentation. Results on a case study demonstrate both theoretical and practical aspects of our proposed methods.

Useful directions for future work include developing a fully-customized sensor node design for our fault tolerant, distributed line-crossing recognition system. Developing a multi-dimensional LCR system, along with appropriate fault tolerance mechanisms, is another promising area for further work.

References

[1] V. Claesson, H. Lonn, N. Suri. An Efficient TDMA Start-up and Restart Synchronization Approach for Distributed Embedded Systems. *IEEE Transactions on Parallel and Distributed Systems*, vol.15, pages 725-739, August 2004.

[2] T. van Dam and K. Langendoen. An Adaptive Energy-efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, November 2003, pp. 171–180.

[3] G. Gupta and M. Younis. Fault-tolerant clustering of wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, vol. 3, pages 1579- 1584, March 2003.

[4] J. Hill and D. Culler, Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, pp. 12–24, November 2002.

[5] M. Kuorilehto, M. Hannikainen, and T. D. Hamalainen. A Survey of Application Distribution inWireless Sensor Networks. *EURASIP Journal on Wireless Communications and Networking*, pages 774-788, 2005.

[6] D. Lymberopoulos and A. Savvide, Xyz: A Motion-enabled Power Aware Sensor Node Platform for Distributed Sensor Network Applications. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, April 2005.

[7] A. Nahapetian, P. Lombardo, A. Acquaviva, L. Benini, and M. Sarrafzadeh. Dynamic Reconfiguration in Sensor Networks with Regenerative Energy Sources. In *Proceedings of the Design Automation and Test Europe Conference*, April 2006.

[8] A. Nisbet and S. Dobson. A systems architecture for sensor networks based on hardware/software co-design. In *Proceedings of the 1st IFIP Workshop on Autonomic Communications*, pages 115-126, Springer Verlag, July 2005.

[9] C. Shen, R. Kupershtok, B. Yang, F. Vanin, X. Shao, D. Sheth, N. Goldsman, Q. Balzano, and S. S. Bhattacharyya. Compact, Low Power Wireless Sensor Network System for Line Crossing Recognition. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, May 2007.

[10] B. Yang, F. Vanin, C. Shen, X. Shao, Q. Balzano, N. Goldsman, and C. Davis. A low profile f-inverted compact antenna (fica) for wireless sensor networks. In *Proceedings of the IEEE AP-S International Symposium*, June 2007. To appear.

[11] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, pp. 493–506, June 2004.