Design and Implementation of Hybrid Circuit/Packet Switching for Wearable Systems

Fardin Derogarian, João Canas Ferreira, Vítor M. Grade Tavares
INESC TEC, Faculdade de Engenharia, Universidade do Porto
{mpt09020, jcf, vgt}@fe.up.pt

***Abstract*—This paper presents a network router and transceiver for wearable, low-power, high-speed Body Area Networks (BAN) applications running in a mesh network of sensors embedded in textiles and connected to each other with conductive yarns functioning as bidirectional transmission channels. The routing of data packets from sensor nodes to a sink node is based on hybrid circuit and packet switching. In comparison with pure packet switching, hybrid routing decreases end-to-end delay, power consumption and buffer size. The proposed design uses independent sender, receiver and circuit switching modules, thereby allowing the nodes to simultaneously send and receive data. The simulation results show that circuit and hybrid switching modes significantly increase the performance of the system. In addition, implementing the complete packet process on FPGA, instead of using an external microcontroller as in previous work, enables a much faster routing process. The results are based on a Verilog description of the system, which has been synthesized for a low-power IGLOO FPGA with Libero Project Manager and simulated with ModelSim. The implementation operates successfully at a data rate of 20 Mbps.**

I. Introduction

Wearable technology, a class of Body Area Networks (BAN), has been developing and attracting many researchers in the past two decades [1], [2]. In particular, wearable systems are often used for health monitoring. These systems can be implemented on wireless, wired or hybrid networks [3]. One of the most important aspects of BAN application is reliability [4]. Many applications are based on wireless networks; but despite its benefits, wireless systems may suffer from interference, fading, and low data rates [3], [5].

In contrast to wireless networks, wired networks can provide high speed and reliable, low-power solutions [6]. Embedded conductors, such as conductive yarns or even copper wires embedded in textiles, provide the electrical connections for the sensor nodes. Using garments with conductive yarns and embedded sensors provides the most comfortable and easiest way to monitor physiologic signals [7], [8]. Although a conductive yarn create an electrical connection, it should be noted that the electrical characteristics of conductive yarns are not exactly those of normal wires. They exhibit significant variation in their electrical properties in relaxed and stretched modes. The differences between conductive yarns and copper wires must be taken into account to achieve a reliable network. Such a network must be tolerant to catastrophic and parametric faults.

BANs are distributed systems in which each sensor acquires data from the body. Many BAN applications also employ sink nodes that collect data from all sensor nodes. The network is usually organized in a bus, star or mesh topology, each with its own merits and shortcomings. When the number of the sensor increases, a mesh network shows better performance. As referred before, reliable communication is important in a BAN application. Due to the conductive yarns vulnerability to the wear and tear, bus or star configurations might be prone to failure, since each sensor shares a single connection. In a mesh network, however, each node usually has more than one connection available for use in case of a failure. Alternative connections increase reliability and system lifetime.

An FPGA-based implementation of the router, which was designed to capture electromyographic signal and movement information from the lower limbs, was introduced in [9] and [10]. The previous approach was based on packet routing with packet switching. The present paper describes and evaluates the design of a new router and transceiver module that supports circuit, packet and hybrid switching for mesh-topology BANs. In comparison to previous developments, the proposed circuit improves the performance of the system by decreasing end-to-end delay, number of memory buffers, and energy consumption. Packet switching is also improved by implementing it inside the router, avoiding the need for an external microcontroller. In addition, the circuit now works with a clock whose frequency is two times the data rate; the previous design required a system clock with four times the data rate. Therefore, for the same reference clock, the data rate of the new circuit is now two times faster than the previous. By reducing the system clock frequency, consequently the energy consumption is also diminished.

The remaining of the paper is organized as follows: Section II describes the related works. Section III describes the routing protocol and the system design. Section IV presents and discusses the results obtained from the simulation of the circuit. Section V sums up the main conclusions drawn from the results.

II. Related Work

In the last years, a number of wired networks for wearable applications have been proposed. Post and Orth built an electronic system by utilizing conductive yarns as a data bus [11]. Based on theoretical analysis and experimental results, the ability of conductive yarn to carry both data and power supply has been shown by Wade and Asada [12]. A wearable personal network (WPN) system based on fabric serial bus with conductive yarns was introduced by Hyung et al. [6]. Their system is a four-layered network using bus topology and implemented on FPGA and works up to 10 Mbps data

978-1-4799-2399-1/14/$31.00 ©2014 IEEE

rate. A host node manages the network and controls communication. Implementation of a near-field coupling transceiver is introduced by Yoo and Lee [13]. The proposed method is integrated with a fault-tolerant network switch for inter-layer and intra-layer wearable BANs and is applicable to multilayer clothes with 10 Mbps data rate.

Given the trend for higher number of sensors and devices and for more complex networks, bus and star topologies may be unable to satisfy the new requirements. Cooperation of sensors in a mesh network is a suitable alternative. Our work has introduced a router for the physical, MAC and network layers of a 4-port network device for mesh networks in wearable applications [9]. The aforementioned circuit is based on routing by packet switching. Although the evaluation of a prototype implementation indicates that the system works fine and handles the communication requirements of a real application [10], [14], the system can still be improved by introducing circuit switching. This paper presents another circuit that supports packet and circuit switching or hybrid switching and with 20 Mbps data rate.

III. Description of the Circuit and Protocol

This section describes the communication protocol for hybrid circuit and packet switching in a mesh network of the Sensor Nodes (SNs) and its complete implementation on FPGA.

A. MAC Protocol

In the wearable systems under consideration, all SNs are connected to each other by conductive yarns. Connection lines are bidirectional and a MAC protocol is used to share them and control the communication. When a SN wants to send data to a neighboring SN, it first checks the line status; if the line is free, it then sends a request message to the other SN. The receiver node may not be ready to communicate, because it may be involved in communicating with another SN. Therefore, an RTS/CTS (Request-To-Send/Clear-To-Send) handshaking mechanism is used to share the line and establish the communication. In this work, the Request to Send (RTS) message is also used to inform the receiver about the kind of packet that is going to be sent.

Figure 1 shows part of a network, including the Base Station (BS) node (the sink node) and regular SNs. We use the Source Routing for Minimum Cost Forwarding (SRMCF) energy-efficient routing protocol described in [15], where a more detailed description can be found. According to this protocol, normal operation must be preceded by a setup phase. In this phase, the routing protocol finds all minimum-cost paths between BS and SNs. The minimum-cost path between node SN3 and the BS in shown with bold lines in Fig. 1. Under SRMCF, each node first determines its neighbor node (called the *near-node*) on the path with minimum cost, then sends the path information to the BS, which keeps a database of available SNs and corresponding paths. In this way, the protocol ensures communication from SNs to BS or in the reverse direction always occurs over paths with minimum cost.

We now describe how a packet from an SN is routed directly to the BS using a circuit created jointly by cooperating MAC and routing protocols.

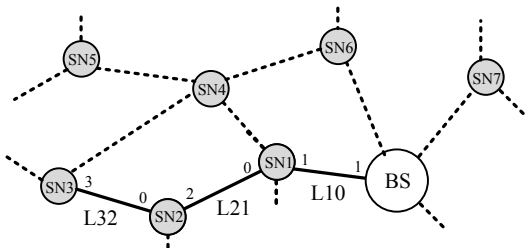

Fig. 1. A mesh network of SNs and BS nodes

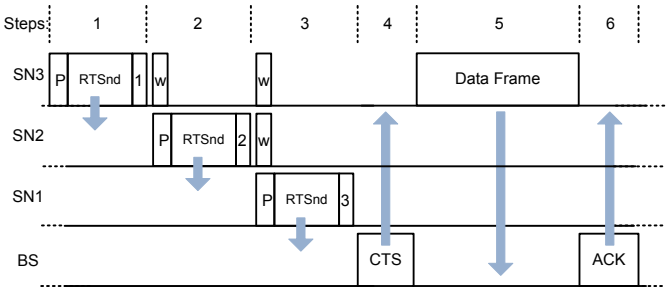

Fig. 2. Routing a packet from SN3 to BS by using circuit switching

Consider the situation where node SN3 wants to send a packet to the BS. The first choice is routing based on circuit switching. Figure 2 shows the network operations as a circuit between SN3 and the BS is established and used to transmit a packet. For that purpose, SN3 sends a request message *RTSnd* to its *near-node* SN2 with hop count 1 (step 1). In Fig. 2, the letter "P" denotes the preamble that it is used for synchronization and waking-up the receiver. *RTSnd* is the RTS message used to establish circuit switching. This message also carries the hop count.

If SN2 is ready, then it sends back a waiting message to SN3 (shown as "w" in the figure) to indicate that it is ready and is going to continue with the process of establishing the circuit by sending a similar request to its own near-node. Then SN2 sends *RTSnd* to SN1 and sets up its own switch so as to connect the lines to nodes SN3 and SN1 (step 2). In this way, any signals from SN1 appear in the line connected to SN3 with very small delay (only due to the signal path between two lines).

Finally, SN1 sends a *RTSnd* request to the BS, and a wait message to nodes SN2 and SN3 (through the partially established circuit) (step 3). It also sets up its own switch module so as to link the ports connected to SN2 and BS (in the same way as is done at node SN2).

After this step, SN3 is connected directly to the BS and receives almost immediately any signals that BS generates. If BS is free to accept, it replies with Clear to Send (CTS) message (step 4). Nodes SN1, SN2 and SN3 receive the *CTS* message simultaneously. Then nodes SN1 and SN2 change the direction of their switches to allow SN3 to start sending packet to the BS (step 5). At the end of packet transmission, nodes SN1 and SN2 change again the direction of their switches to hand over the *ACK* message from BS to SN3 (step 6). After that all nodes end the communication and release the lines.

If one of the intermediate nodes is not ready, then circuit-

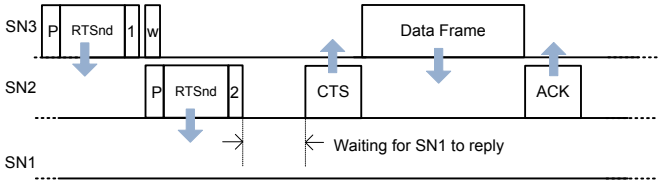

Fig. 3. Packet switching between SN3 and SN2

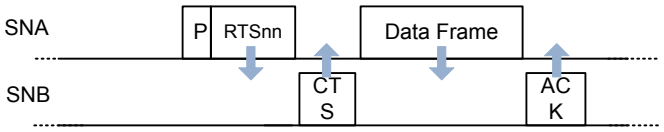

Fig. 4. One-hop packet handover

based communication cannot be done as just described. In this case, two options are possible: 1) releasing the partially established circuit path, waiting for a while and then retrying; 2) having the last node, before the busy node, receive the packet, waiting for the next node to be free and then send the packet on.

The second option means using a hybrid routing mode based on both circuit and packet switching. The hop count carried by the *RTSnd* provides information about the number of intermediate nodes, and can be used by the last node to choose between options 1 or 2. The node that takes this decision will also have in consideration its own status (like the currently available memory for storing the packets).

Figure 3 illustrates the situation when SN1 is not ready and SN2 decides to accept the packet from SN3 and hand it over later: SN2 sends a *CTS* message to SN3 and starts to receive the packet from SN3.

For the most part, traffic in sensor networks is due to data transmissions from sensor nodes to BS. The SRMCF protocol also supports broadcasting, base-to-node and node-to-node communications. For these transmissions packet switching is always used as shown in Fig. 4: the sender node sends a *RTSnn* message to the next node and packet transmission starts if the receiver node replies with a *CTS* message. *RTSnn* messages do not carry any extra information.

B. Router and Transceiver Implementation

Figure 5 shows the block diagram of the implemented transceiver with the sub-modules for handling the physical, MAC and network layers. The transceiver has a 2 KB SRAM to buffer the packets and an SPI port interface to connect to a microcontroller. The modules TX and RX are responsible for sending and receiving packets, respectively. Module CSW handles circuit communication. As mentioned before, the use of circuit or packet switching is determined by *RTSnd* and *RTSnn* messages in the MAC layer. Module SD detects the type of incoming *RTS* message and selects the CSW or RX modules based on the *RTS* message. Module LINE STATUS determines whether the line is free or not. Module TX LINE SWITCH is responsible for creating signals in physical layer and sending the data to other nodes. Because modules TX, RX and CSW

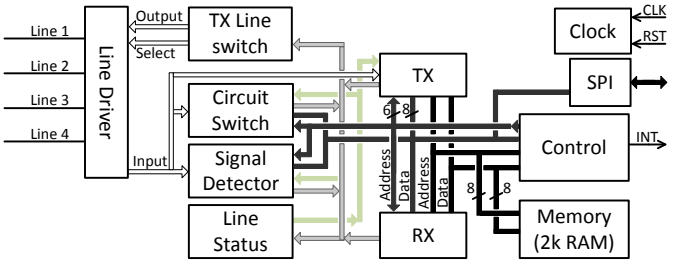

Fig. 5. Block diagram of implemented transceiver

work independently, the circuit can send and receive packets simultaneously (over different lines).

The CONTROL module uses two 8-bit internal buses to access the registers in the TX and RX modules; another data bus is used to access the packet buffer. The other buses are used for intra-module communication. All the registers are accessible via the SPI port connected to an external microcontroller. The CONTROL module is responsible for controlling the register bus and the read/write operations from/to the registers. For that, the CONTROL module first receives a command from the SPI module specifying the type of operation and the address of the register; it then allows to SPI module to connect directly to the register and continue the process.

The MEMORY module includes a 2 KB SRAM to buffer the packets and additional sub-modules to allocate the memory and control the data bus. It shares the buffer and the data bus with the RX, TX and CONTROL modules. If a module wants to read or write data form the buffer, it first sends a request to the MEMORY module, which then allocates a time slot for it to access the bus. The external microcontroller accesses the buffer via the CONTROL module for reading or writing packets.

1) Signal Detector: As mentioned, *RTS* messages not only indicate a request to establish communication, but also carry information about the type of the packet. Figure 6 shows the block diagram of the SD module. For each line, there is a RTS DETECTOR sub-module, which detects the preamble and then starts to determine the type of *RTS* message. Based on the *RTS* message, this module selects the RX or CSW modules if they are free. It also generates a synchronized clock and detects the incoming data. After that, the RX or CSW modules continue with the communication. If no modules are free, the RTS DETECTOR drops the received message and waits for the next incoming message. After ending the communication, the RX or CSW modules send a signal to the SD to indicate the end of the communication, so that SD is ready for the next packets. Because all the SNs work independently, it is possible that two or more SNs may start to communicate with the same SN receiver, simultaneously. If they also send the same *RTS* message, the RTS DETECTOR in the receiver might select modules RX or CSW simultaneously. To avoid such a situation, a timing sub-module in SD shares the access time of RTS DETECTORS between the RX and CSW modules.

2) Transmitter: Figure 7 shows the packet transmitter module. If there is a packet in the buffer, ready to be sent, the TX module first checks the status of the destination line, and starts to send a *RTS* message to the receiver SN if the line is free. Then, the TX module waits for a *CTS* message (detected

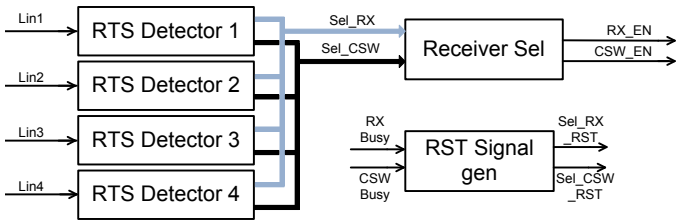

Fig. 6. Block diagram of signal detector

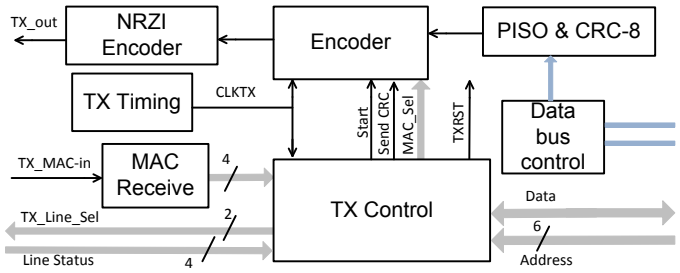

Fig. 7. Block diagram of transmitter

by the MAC RECEIVER sub-module). Having its own module to receive MAC messages makes the TX module independent of the RX module. Therefore, circuits can send and receive packets simultaneously from different SNs. When the MAC RECEIVER sub-module detects the *CTS* reply, communication is started by sending the packet to the receiver SN via the selected line. The MAC RECEIVER also detects the *ACK* message at the end of communication. The ENCODER unit includes all sub-modules for generating *RTS* messages, including data encoding and synchronization bits. The input and output of this module is serial, supplied by a PISO (parallel-in serial-out) converter, whose input is connected to the buffer, and whose output is connected to the NRZI line encoder. The trailer part of the frame is a CRC-8 checksum, which is used to detect errors that may occur during transmission.

3) Receiver: Figure 8 shows the receiver module. If the SD module detects a *RTS* message and selects the RX module, then RX starts to send a *CTS* message to the sender node. To be independent from the other modules, RX has a sub-module to generate MAC messages. All the input signals of RX module come from SD. Therefore, modules RX and SD collaborate in the reception of a packet. After sending the *CTS* reply, packet receiving starts by detecting the signals that are necessary to synchronize the first bit of the packet via the FRAME SYNC sub-module. The DECODER module is responsible

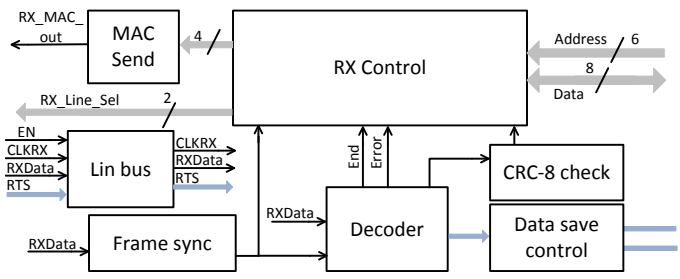

Fig. 8. Block diagram of receiver

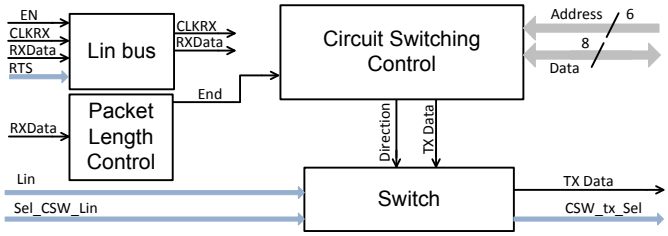

Fig. 9. Block diagram of circuit switching

for receiving and decoding the incoming data. The RX module has access to the buffer for saving received packet via the DATA SAVE CONTROL sub-module. At the end of the packet and if the CRC-8 CHECK sub-module indicates that no errors occured, the RX module sends the *ACK* message to the sender, finalize the received packet and releases the line. Otherwise, it sends the *Error* message to the sender, drops the packet, and releases the line.

4) Circuit Switch: If a SN wants to send data to the BS, the first choice is circuit switching. Module CSW, shown in Fig. 9, is responsible for establishing the circuit between input and output lines, and for handling the communication while circuit switching is being used. After receiving *RTSnd* message, the SD module directs the next part of the message to the CSW, which receives the hop count. Then the CSW module sends a waiting message to the sender, increases the received hop count value by one, creates another *RTSnd* message, and sends it to the line connected to the near-node. The sub-module SWITCH connects the lines directly together to make a circuit connection. The direction of the switch is controlled by the CIRCUIT SWITCHING CONTROL sub-module. During communication, the CSW module monitors the packet length to detect the end of the packet, and to change the direction of the switch to allow *ACK* message to pass to the sender of the packet. In two cases the circuit switch (CSW) module may choose to receive packets via the RX module, instead of creating a circuit path: 1) When its near-node is busy (absence of CTS message); 2) when the hop count value reaches the maximum allowed predefined value. The second option lets the user (or upper-level layers) manage the network based on the application requirements. In this case, CSW directs communication to the RX module to start receiving the packet. The received packet will be buffered and sent later to the BS.

IV. DESIGN VERIFICATION

The router has been implemented on the IGLOO AGLN0250 FPGA, a low power device from Actel. Previous implementations supporting only packet switching [9], [10] used an IGLOO AGLN0125, which is a similar device with fewer logical cells. The circuit was described in Verilog and synthesized with Libero Project Manager V9.1 from Actel. The resulting implementation uses 5137 logic cells (84 % of the available cells). The built-in memory of the FPGA is used for buffering the packets. The synthesized circuit works at 40 MHz to have 20 Mbps data rate. To simulate the circuit, a network of SNs and BS, as shown in Fig. 1, was used. Each SN has an additional microcontroller, which is connected to the transceiver via the SPI port. The microcontroller is used

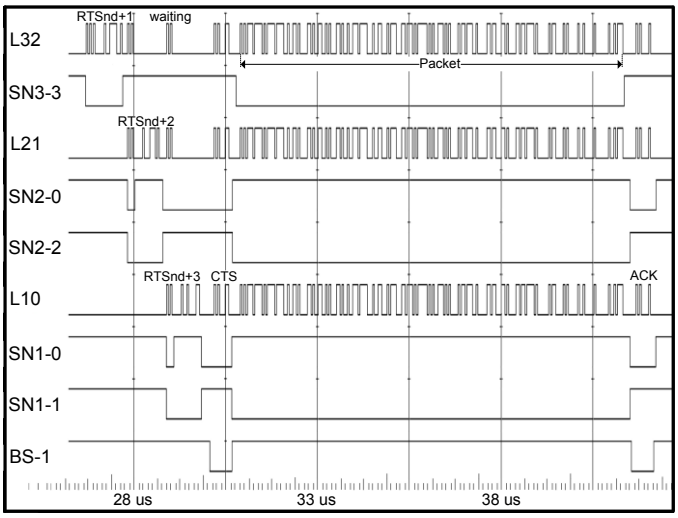

Fig. 10. Routing a packet from node SN3 to the BS using full circuit switching

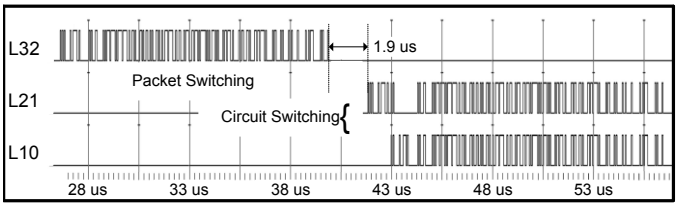

Fig. 11. Hybrid routing combining circuit and packet switching.

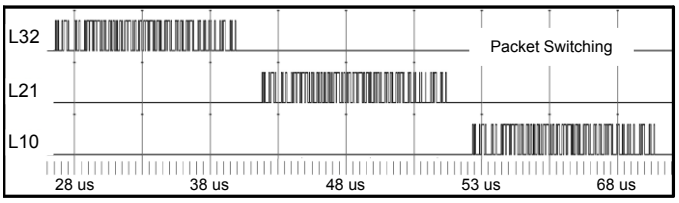

Fig. 12. Routing using only packet switching.

for configuration and to read/write data from/to the packet buffer. The test-bench includes some microcontroller modules to generate the appropriate signals for the SPI connection.

Figure 10 presents the simulation results corresponding to routing a packet from SN3 to the BS as described in Sec. III-A. The intermediate nodes are SN2 and SN1, respectively and routing is based on circuit switching. This figure also shows the status of the SNs ports, e.g., SN2-0 indicates port 0 of SN2. The low level (zero) indicates that the node is transmitting and a high level indicates receiving or listening status. Node SN3 starts communication by sending *RTSnd + 1* to SN2 at $t = 26.7\,\mu s$. Node SN2 receives the *RTS* message, sends a *Wait* message to SN3 and *RTSnd + 2* to SN1 simultaneously at $t = 27.85\,\mu s$. SN1 starts the communication with the BS at $t = 28.9\,\mu s$. After this step, a circuit path has been formed between BS and SN3. The path is setup for communication from BS to SN3. Therefore, SN3 receives the *CTS* message from BS directly. Then, SN2 and SN1 change the direction of the switch to allow SN3 to transfer the packet starting at $t = 30.9\,\mu s$. The packet has 22 byte and takes $10.6\,\mu s$ to transmit, finishing at $t = 41.5\,\mu s$. A small packet was selected to clearly show the message in figure. At the end of the packet, SN2 and SN1 change the direction of their switches to transmit the *ACK* message from the BS. All nodes release the lines and finish the communication at $t = 42.3\,\mu s$. Therefore, the end-to-end delay is $15.6\,\mu s$. Figure 11 shows the result of routing the same packet via hybrid packet and circuit switching. Node SN3 starts to send the packet to SN2 at $t = 26.7\,\mu s$. This step takes $13.4\,\mu s$ and finishes at $t = 40.1\,\mu s$. Node SN2 saves the received packet in its internal buffer. The routing service at node SN2 takes $1.9\,\mu s$ to process the packet and starts to route the packet to the BS direction. In this step, routing is based on circuit switching, starting at $t = 42\,\mu s$ and finishing at $t = 56.7\,\mu s$. Therefore in this case, end-to-end delay of routing a packet of length 22 byte takes $30\,\mu s$, that is $14.4\,\mu s$ more than the full circuit switching with the same packet size. In our previous work [10], the packet routing service time was around $150\,\mu s$, almost 80 time more than the value obtained in this work, because it required software processing by the external microcontroller (the MAC layer was implemented in software). The speedup in routing service time of the present implementation is due to performing the entire packet processing in the FPGA without involving the microcontroller.

Figure 12 shows the transmission of the same packet using just packet switching. The routing starts at $t = 26.7\,\mu s$ and finishes at $t = 72\,\mu s$. In comparison with the previous cases, it takes respectively $29.7\,\mu s$ (70%) and $15.3\,\mu s$ (27%) longer to deliver the packet to the BS.

The worst-case situation for routing occurs for networks in which SNs are arranged in a line. Assume that the BS node is at the end of the line and the first SN is sending packets (22 byte) to the BS as is shown in Fig. 13. Figure 14 summarizes the simulation results for the end-to-end delay in networks with 5, 10 and 15 SNs. The abscissa is the number of nodes using packet switching; the other nodes constitute a single circuit connection. Abscissa 0 corresponds to the situation of a single circuit from the leftmost node to the BS. It can be observed that that circuit switching decreases the end-to-end delay. For example, with 15 SNs, the delay using only packet switching is $228.75\,\mu s$, that is $7.8\times$ larger than the delay when only circuit switching is used. The delay ratio in comparison with networks including 5 and 10 SNs is 4.27 and 6.46, respectively. Figure 14 also shows that the delay

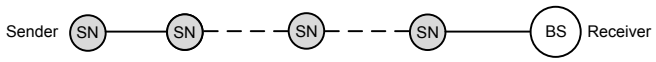

Fig. 13. A network of SNs and BS in line topology.

difference between networks with 5 SNs and 15 SNs is $11.5\,\mu s$ when full circuit switching is used, whereas the difference is $152.5\,\mu s$ for full packet switching. Figure 15 shows the ratio of maximum (for packet switching) and minimum (for circuit switching) end-to-end delay for the same network shown in Fig. 13 with up to 200 nodes in series.

We may conclude that circuit switching significantly limits the end-to-end delay when the number of SNs increases. Because there is no need to buffer the packets in circuit switching, the buffer size is also reduced when full circuit

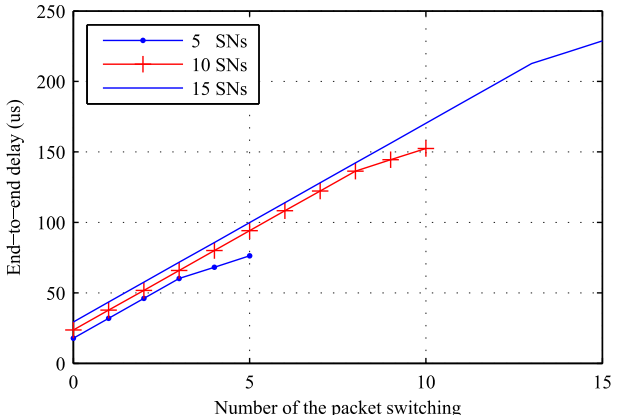

Fig. 14. End-to-end delay for hybrid switching as a function of the number of nodes using packet switching.

switching or hybrid switching is used. Buffering and packet processing consume more energy than just switching.

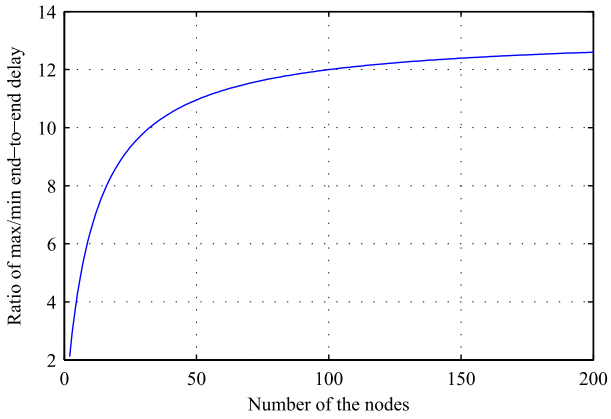

Fig. 15. The ratio of maximum to minimum end-to-end delay for a line topology as a function of the number of nodes.

V. Conclusion

This paper describes a transceiver and router modules designed for wearable systems. The FPGA implementation of the module works as expected in a prototype mesh network of sensor nodes connected by conductive yarns. Routing of packets from SNs to BS is performed completely on-chip, so that the end-to-end delay is significantly smaller than the one achieved by implementations that use an external microcontroller to handle the routing protocol (as was the case of a previous implementation of the same protocol for pure packet switching discussed in [9]).

Routing is based on a combination of circuit and packet switching. The correct operation has been verified by simulation and by a prototype implementation on FPGA. Simulation results show that circuit switching decreases end-to-end delay significantly. It also decreases the number of packets buffered: with circuit switching intermediate nodes route the packet without any buffering or processing, an important aspect for systems with limited resources. As a consequence, packet switching consumes more energy than the circuit switching; therefore, network lifetime increases when using circuit switching. The use of a hybrid routing method increases flexibility and leads to better network performance. Depending on the network topology and higher-level protocols, the circuit can be configured to work in any of the supported modes (circuit switching, packet switching, or a combination of both).

Acknowledgments

This work is financed by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project PROLIMB PTDC/EEA-ELC/103683/2008. and through Ph.D. grant SFRH/BD/75324/201.

References

[1] Y. Meng and H. C. Kim, "Wearable systems and applications for healthcare," in *First Intl. Conf. on Computers, Networks, Systems and Industrial Engineering (CNSI)*, 2011, pp. 325–330.

[2] S. Park and S. Jayaraman, "Smart textile-based wearable biomedical systems: A transition plan for research to reality," *IEEE J. on Information Technology in Biomedicine*, vol. 14, no. 1, pp. 86–92, 2010.

[3] J. Xing and Y. Zhu, "A survey on body area network," in *5th IEEE Intl. Conf. on Wireless Communications, Networking and Mobile Computing*, Sep. 2009, pp. 1–4.

[4] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. Leung, "Body area networks: A survey," *J. Mob. Netw. Appl.*, vol. 16, no. 2, pp. 171–193, Apr. 2011.

[5] I. Howitt and J. A. Gutierrez, "IEEE 802.15.4 low rate - wireless personal area network coexistence issues," in *IEEE Wireless Comm. Network.*, vol. 3, Mar. 2003, pp. 1481–1486 vol.3.

[6] H. S. Lee, C. B. Park, K. J. Noh, J. Sunwoo, H. Choi, and I.-Y. Cho, "Wearable personal network based on fabric serial bus using electrically conductive yarn," *ETRI J.*, vol. 32, no. 5, pp. 713–721, Oct. 2010.

[7] A. Lymberis and L. Gatzoulis, "Wearable health systems: from smart technologies to real applications," in *28th IEEE Intl. Conf. on Engineering in Medicine and Biology Society, EMBS06*, vol. Supplement, Sep. 2006, pp. 6789–6792.

[8] L. Gatzoulis and I. Iakovidis, "Wearable and portable eHealth systems," *IEEE J. on Eng. in Medicine and Biology Magazine*, vol. 26, no. 5, pp. 51–56, Oct. 2007.

[9] F. Derogarian, J. C. Ferreira, and V. M. G. Tavars, "Design and implementation of a circuit for mesh networks with application in body area networks," in *Fifth Euromicro Conf. on Digital System Design, DSD2012*, September 2012.

[10] F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, "Using a wired body area network for locomotion data acquisition," in *27th Conf. on Design of Circuits and Integrated Systems, DCIS2012*, November 2012.

[11] E. R. Post and M. Orth, "Smart fabric, or wearable clothing," in *First Intl. Symp. Wearable Computers*. IEEE, Oct. 1997, pp. 167–168.

[12] E. Wade and H. Asada, "Conductive fabric garment for a cable-free body area network," *IEEE J. on Pervasive Computing*, vol. 6, no. 1, pp. 52–58, 2007.

[13] J. Yoo, S. Lee, and H. Yoo, "A 1.12 pJ/b inductive transceiver with a Fault-Tolerant network switch for Multi-Layer wearable body area network applications," *IEEE J. on Solid-State Circuits*, vol. 44, no. 11, pp. 2999–3010, Nov. 2009.

[14] A. Zambrano, F. Derogarian, R. Dias, M. J. Abreu, A. Catarino, A. M. Rocha, J. M. da Silva, J. C. Ferreira, V. M. G. Tavares, and M. V. Correia, "A wearable sensor network for human locomotion data capture," in *9th Intl. Conf. Wearable Micro and Nano Technologies for Personalized Health*, jun 2012.

[15] F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, "A routing protocol for WSN based on the implemention of source routing for minumum cost forwarding method," in *Fifth Intl. Conf. Sensor Tech. Appl. SENSORCOMM2011*, J. M. Hovem, J. Ellul, L. Gomez, and Y. Reddy, Eds. ThinkMind, Aug. 2010, pp. 85–90.