Energy-Efficient Reconfiguration of Flash-based FPGAs in Heterogeneous Wireless Sensor Nodes

Andreas Engel and Andreas Koch

Embedded Systems and Applications Group (ESA)
Technische Universität Darmstadt, Germany
{engel,koch}@esa.tu-darmstadt.de

***Abstract*—Field Programmable Gate Arrays (FPGAs) can be used as hardware-accelerators in Wireless Sensor Networks (WSNs). Their in-field reconfigurability can be utilized for bug fixing and capability enhancements, but the distributed nature of WSNs strongly demands Over-the-air Programming (OTAP) for reconfiguration to avoid manual maintenance at every single network node. In this paper, the implementation of OTAP for Microsemi IGLOO FPGAs utilized in heterogeneous WSN motes is described. As the entire configuration bitstream can not be buffered at the sensor node at once due to memory limitations, a paging mechanism is utilized and improved to minimize the energy-intensive radio network traffic required for the bitstream configuration and verification. By applying lossless bitstream compression, the overall configuration time can be reduced by 11 % to 24 %, depending on the complexity of the configured hardware design. The energy spent during the configuration process is reduced by 11 % to 37 % using this compression. Another approach utilizes the prediction of page requests to overlap the wireless transport of the next page with the processing of the current page. This prediction mechanism reduces the configuration time by 40 %, independently of the actual bitstream content. The required configuration energy is also reduced by 40 % using the page prediction.**

I. Introduction

In the last years, more and more FPGAs are utilized in WSNs as hardware-accelerators for energy-efficient data aggregation [1], [2], encryption [3], [4], or the realization of complex routing protocols [5]. Besides the lower non-recurring engineering cost of FPGAs, most research groups avoid application-specific integrated circuit hardware-accelerators to keep their WSN platforms flexible after deployment. The flexibility to reprogram the FPGA on-site can be exploited to fix bugs in the post-laboratory development stages, or to adapt the WSN's capabilities to recent encryption and communication standards. Some authors also propose Partial Dynamic Reconfiguration (PDR) to frequently and regularly switch between different hardware-accelerated algorithms on small FPGAs [6], but this work is focused on irregular and unplanned reconfigurations of complete bitstreams, that can not be generated and stored on the WSN mote before deployment.

A WSN typically consists of many nodes distributed over a large area. In some applications, the nodes are also hard to reach for human operators. In all cases, being able to reconfigure the nodes via the wireless infrastructure greatly improves the maintainability and flexibility of the network. This process is referred to as OTAP and requires some specific hardware features. First, the communication protocol must be able to actually transport the new bitstream to a specific or all WSN nodes. Second, the WSN nodes must be able to receive and buffer or immediately process the bitstream. The buffer memory might be located outside the FPGA, from where it can be loaded after resetting the reconfigurable hardware. This is the typical use-case for FPGAs with a Static Random Access Memory (SRAM) configuration storage, and it does not require an FPGA-external Microcontroller Unit (MCU) to manage the reconfiguration process. Alternatively, the received bitstream can be stored directly into the configuration memory of the FPGA. In this case, the bitstream reception overlaps with the actual reconfiguration, and the entire process has to be controlled by an FPGA-external MCU. The latter must have access to the programming interface (e.g., a Joint Test Action Group (JTAG) port) of the FPGA. In both scenarios, if the targeted bitstream buffer is a Flash-based Non-Volatile Memory (NVM), the sensor node must be able to generate the voltage required to modify the Flash memory, which is typically higher than the voltage required to only read the NVM. And finally, the limited energy buffer as well as the constrained computational resources of the low-power sensor nodes must be able to support the entire bitstream transport and configuration process.

This paper targets the second scenario, i.e., the reconfiguration of an FPGA by a dedicated MCU, which receives the configuration bitstream wirelessly from a gateway node, as shown in Fig. 1. More concrete, heterogeneous wireless sensor

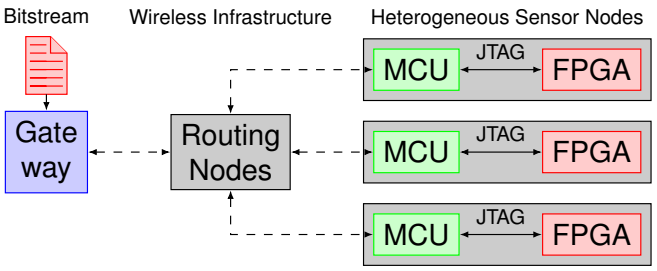


Fig. 1. Targeted OTAP scenario: Wireless bitstream transport from a gateway to the MCU of one (or many) heterogeneous sensor node(s), from where the bitstream is written into the configuration memory of an FPGA using a JTAG interface.

978-1-4673-9406-2/15/$31.00

nodes based on Microsemi IGLOO low-power FPGAs such as the HaLOEWEn [1], the PowWow [7], or the low power Cookie mote [8] are of special interest. The IGLOO device family is based on a Flash configuration memory accessible by a JTAG port. This paper further assumes that there is no memory available on the sensor node sufficiently large to store the entire configuration bitstream at once. Unencrypted IGLOO bitstreams for the AGL1000 device utilized by the HaLOEWEn mote have a constant size of 915 kB independent from the actual hardware design, when stored in the format required for direct JTAG programming. The MCU internal SRAM is typically much smaller, and additional external low-power memory is not available as single 1 MB SRAM module with a serial interface. So, either multiple smaller SRAM modules with an inter-integrated circuit interface would have to be mounted on the sensor node (e.g., $8 \times 1\,\mathrm{Mbit}$), which would occupy a large area on the Printed Circuit Board (PCB) of the WSN node (e.g., 60 mm^2 for eight small outlint packages). Alternatively, a single large SRAM module with a parallel memory interface would have to be connected to the MCU, which would require at least 21 General Purpose Input/Output (GPIO) pins for a $1024 \times 8\,\mathrm{bit}$ module. As both, the PCB area and the MCU GPIO pins are restricted resources in WSN motes, not being able to buffer the entire bitstream at once is a realistic restriction.

Microsemi provides a software library called DirectC [9] enabling the use of MCUs as JTAG programmers for Microsemi devices. DirectC already supports a paging mechanism used to load and buffer the bitstream in smaller junks. However, DirectC is not optimized for OTAP applications, where the number of page accesses has to be minimized to reduce the network traffic and thus the overall energy spent on the bitstream transport. The main contributions of this paper are therefore,

- the description of a wireless communication protocol supporting the existing DirectC paging mechanism,
- the reduction of the number of wireless packets to be transmitted during bitstream programming and verification by means of page compression and prediction,
- the evaluation of the achieved improvements in terms of reduced time and energy spent for the configuration process on the HaLOEWEn mote.

The remaining parts of this paper are organized as follows. Section II provides an overview of related research efforts for OTAP of FPGAs. Section III describes the basic paging mechanism of DirectC. The OTAP-related drawbacks of DirectC are outlined in Section IV. Furthermore, this section details the basic concepts of a wireless bitstream transfer protocol as well as the improvements applied to minimize the required network traffic. Section V evaluates the achieved reduction of the configuration time and energy, while Section VI concludes this work with some suggestions for further improvements.

II. Related Work

A number of wirelessly reconfigurable heterogeneous sensor nodes with FPGA-based hardware accelerators have been reported, such as [10], [11], [12]. However, those authors did not report details about the achievable reconfiguration performance such as the required time or energy, and can therefore not be compared to the current work. The following analysis is thus focused on related work actually reporting their required OTAP duration.

A Programmable System-on-Chip (PSoC) extension board for an IEEE 802.15.4 transceiver-based WSN mote was proposed in [13]. A dedicated complex programmable logic device and a large configuration storage was utilized for the OTAP of a Cypress PSoC hardware-accelerator. The authors report a configuration duration of 16 s, including 1 s for the wireless transfer of the 32 kB bitstream.

The Cookie node described in [14] also consists of an 8 bit MCU and an IEEE 802.15.4 transceiver, but it utilizes a Xilinx Spartan-3 FPGA as its hardware-accelerator. The authors evaluated the time and energy required for the PDR of the FPGA. The reconfiguration of six configurable logic blocks (i.e., 8.7 kB configuration data) took 118 s and consumed 36.5 J.

The WSN mote proposed by [15] consists of a 16 bit MCU, a low-power short range device transceiver and a Lattice iCE40 hardware-accelerator. The authors report 200 s to be required for the entire OTAP procedure. The size of the bitstream is not specified, but the Lattice device (iCE40LP1K) provides nearly $20\times$ less logic resources than the Microsemi AGL1000 used throughout this work.

As will be shown in Section V, the relative configuration time per configuration bit achieved throughout this work outperforms the systems described above.

A platform independent OTAP improvement was proposed in [16]. The authors observed that for typical firmware updates, the new configuration will only slightly differ from the previous one. So a dictionary-based delta compression algorithm can significantly reduce the required network traffic. As the authors evaluated their concept using powerful processors and transceivers typically used in mobile phones, their results can not be easily transferred to the WSN domain.

Besides [14], no other research described above actually analyzed the energy consummation of the reconfiguration process. Furthermore, all authors strictly separated the bitstream transfer from the configuration process. The wireless paging strategy and the corresponding energy consumption analysis described in this work thus significantly extends the existing literature.

III. DirectC Software Library

DirectC [9] is a software library provided by Microsemi. It is used to let an MCU control the JTAG port of a Microsemi FPGA device in order to program the bitstream provided in the proprietary binary DAT format is loaded into the FPGA-internal configuration memory. DirectC is configurable by preprocessor directives such that only the features required for the selected device (e.g., NVM programming, encryption, sending debugging messages, bitstream paging) are included in the compiled firmware. This is essential to reduce the memory

footprint of the JTAG library, which has to fit in the memory constrained MCU next to the actual MCU application code.

Microsemi also provides an alternative JTAG programming library called STAPL Player [17]. It is intended for generic JTAG programmers supporting multiple device families at runtime, as the actual device-specific programming algorithm is encoded in the bitstream files. This flexibility is not required for the OTAP of a specific FPGA included in a heterogeneous WSN mote. Furthermore, Microsemi suggests to use DirectC for resource constrained MCUs [9]. Therefore, the STAPL Player is not considered in this work.

When reconfiguring a Flash-based IGLOO FPGA, DirectC is working in six phases. In the initial *sanity check phase*, a header section within the bitstream is loaded to check whether the bitstream is actually targeting the device type attached to the JTAG port. Furthermore, a 16 bit Cyclic Redundancy Check (CRC) sequence of the entire bitstream is calculated by DirectC and compared to the checksum stored in the bitstream. If both consistency checks were passed, the FPGA is prepared for the subsequent steps in the *setup phase*. Afterwards, the entire configuration memory is deleted in the *erase phase* and rewritten in the following *write phase*. The bitstream written to the configuration memory is compared against the provided bitstream in the *verification phase*. Finally, the FPGA is prepared for deployment in the final *release phase*.

To port DirectC to a specific MCU, several functions have to be implemented for delaying the program execution, accessing the GPIO pins connected to the FPGA JTAG port, displaying debug messages, and loading bitstream pages into the MCU-internal memory. The latter (named `dp_get_page_data`) is called with a single parameter, i.e., the start address of the requested page. This page load function is the hook to the wireless network. In the baseline implementation of the proposed OTAP protocol described in Fig. 2, a page request is sent by the DirectC-executing MCU to the gateway node using the existing routing infrastructure of the WSN application. The gateway node extracts the requested page from the overall bitstream and sends the page back to the requesting sensor node. Afterwards, DirectC analyzes the received page and programs the relevant bits into the FPGA using the appropriate JTAG commands. This page load and evaluation process is repeated, until an error occurred in the verification phase, or the FPGA was successfully reconfigured. The corresponding exit code is sent back to the gateway node, as shown in Fig. 2.

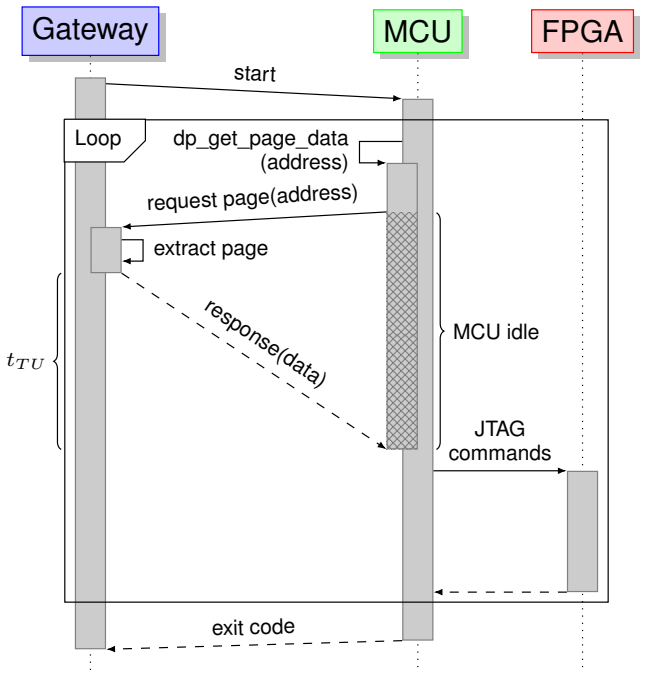


Fig. 2. Baseline OTAP page loading procedure.

IV. Improved Wireless Pageing

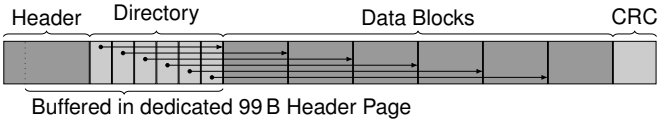
In this section, a list of improvements for the baseline DirectC-based OTAP protocol described in Section III is proposed.

A. Relocation of the checksum calculation

DirectC is requesting the entire bitstream three times during the configuration process, i.e., in the sanity, the write, and the verification phase. The write phase is obviously required. The verification phase is vital to detect write errors, which might be caused by unstable supply voltages during the erase and write phase. The CRC-calculation in the sanity phase however is only required to validate the bitstream file as such. There is no need to perform this validation on the wireless sensor node, so it can be relegated back to the gateway thus reducing the overall network traffic by roughly 33 %. Once validated, the checksum mechanism of the wireless communication infrastructure itself ensures the integrity of the bitstream pages on their way from the gateway to the destination sensor node.

B. Reduction of repeated page transfers

In addition to the two remaining phases requiring to load the entire bitstream (i.e., write and verification), some DirectC-related peculiarities cause specific pages to be requested multiple times even within one phase. As shown in Fig. 3,



Block ID	Address	Length	Description
1	123	9 B	Design name
2	132	174 B	Boundary scan pattern
3	306	4 B	Encryption key
4	310	4 B	Datastream checksum
5	314	936 000 B	Datastream
6	396314	208 B	Lock configuration

Fig. 3. DirectC DAT format consisting of a header, a directory, a fixed number of data blocks, and a CRC checksum. AGL1000 bitstreams always contain the six specified data blocks.

the DAT file format is organized as a 69 B header followed by a directory, a number of data blocks, and a 16 bit CRC footer. The directory describes the type, position, and size of the data blocks with 9 B per entry. All DAT bitstreams for AGL1000 devices contain the six data blocks specified in Fig. 3. As the header and the directory is required several times throughout the entire configuration process, a dedicated node-local buffer for both structures is managed next to the actual page buffer. As the first 24 B of the header are not actually relevant for the configuration, a $69\,\mathrm{B} - 24\,\mathrm{B} + 6 \cdot 9\,\mathrm{B} = 99\,\mathrm{B}$ buffer is sufficient for this purpose. After the first header access, subsequent accesses to this DAT file region do not require an explicit wireless page request anymore.

Another source of inefficiency within the paging mechanism provided by DirectC is the request of a new page, as soon as one of the last 16 B within the current page is accessed. At first, this sounds like a reasonable preloading strategy suitable to exploit the linear bitstream access pattern mainly used throughout the configuration process. However, the remaining 16 B of the current page are not actually processed while the next page is loaded. Instead, DirectC stalls until the next page is available, thus effectively reducing the available page size. Even worse, the loaded pages overlap by 16 B as shown in Fig. 4, thus increasing the overall network traffic. This inefficient preloading behavior was thus disabled to improve the wireless paging mechanism.

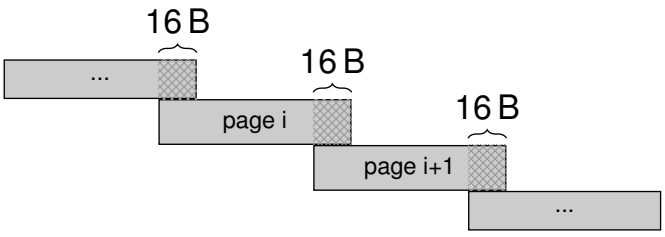


Fig. 4. Overlapping pages in original DirectC implementation. The crossed memory is never read.

C. Page Compression

As stated in Section I, the size of the bitstreams in the DirectC DAT format is not affected by the actual hardware design. This indicates a large amount of redundancy and thus a high potential for a lossless bitstream compression, at least for designs with a low resource utilization. By reducing the number of bytes required to represent a specific bitstream page, the time and energy required to wirelessly transmit the compressed page is also reduced. However, the time and energy required to decompress the pages at the resource constrained sensor nodes also has to be considered.

To exploit this potential, three compression schemes have been implemented, namely Run-Length Encoding (RLE), Run-Length Encoding with Marker (RLEM), and a static Huffman codec [18]. More complex dictionary-based compression schemes are not suitable for this OTAP application, as the transmitted pages are relatively small (about 100 B as limited

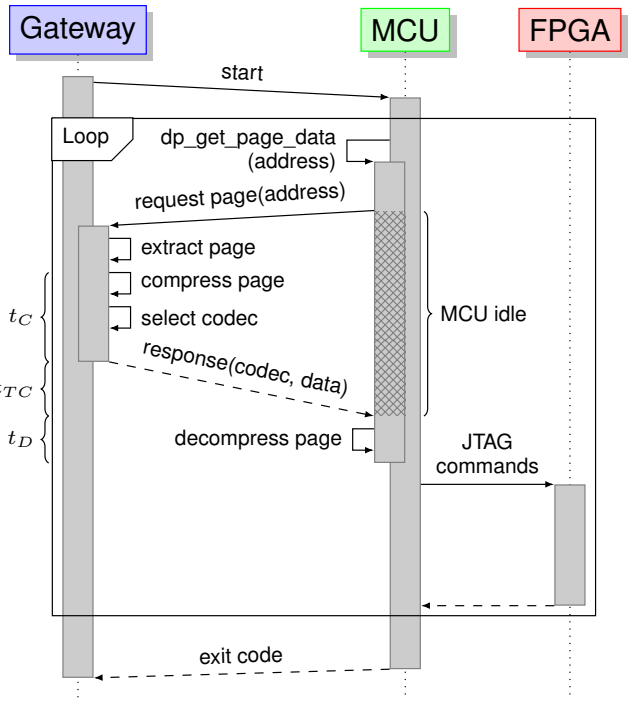


Fig. 5. OTAP page loading procedure with compressed pages.

by the wireless payload), and the dictionaries themselves would exceed the memory capabilities of the MCU.

RLE replaces every sequence of repeated bytes b^n by a pair of bytes $(n, b) \in \{0, \ldots, 255\}^2$ with $n > 0$. As RLE allows a run-length of $n = 1$, the compressed page may even be larger than the uncompressed page. This deficiency is mitigated by the RLEM scheme by applying an escape mechanism. Only for $n > 1$ or $m = b$ the byte sequence is replaced by a triple $(m, n, b) \in \{0, \ldots, 255\}^3$ marked with a rarely occurring symbol m. The analysis of some sample bitstreams yielded $m = 87$ to be most appropriate. Finally, the more sophisticated Huffman codec assigns bit sequences of different lengths to each byte of the uncompressed page, such that more frequently occurring bytes are represented by shorter bit sequences. Instead of deriving bitstream- or page-specific Huffman codes, a static bit sequence-assignment matching the statistical properties of two sample bitstreams (denoted as D1 and D2 in Section V) is used by the Huffman codec. The decoders for all three compression schemes were implemented on and optimized for the resource constrained MCU.

Fig. 5 details the integration of the compression scheme into the page loading procedure. After the requested page was extracted from the bitstream at the gateway node, all available compression schemes are applied. The codec with the best compression ratio for the specific page is selected and reported back to the requesting sensor node along with the compressed page. If none of the implemented compression schemes actually reduces the page size, it is transmitted uncompressed. The execution time t_C of the encoder can be neglected, as it is performed on the gateway node with unconstrained memory and computational resources. However, the page decompression is

executed on the low-power sensor node. To realize a benefit from the page compression, the additional time t_D spent for decoding the page must be smaller than the reduced time spent for the page transmission, i.e., $t_{TU} > t_C + t_{TC} + t_D$. Here, t_{TU} and T_{TC} are the transmission durations for the uncompressed (see Fig. 2) and compressed (see Fig. 5) pages.

D. Page Prediction

As shown in Figs. 2 and 5, the DirectC-executing MCU has to be stalled until the requested page was received. To further speedup the configuration process, these idle times have to be eliminated by already requesting the next page while processing the current page. To request the next page from the gateway before DirectC actually provided the address of the next required page, the page access pattern must be reliably predictable.

Fig. 6 shows the page access pattern during the entire configuration process for the maximum supported page size derived in Section V after eliminating most of the repeated page requests, as described in Section IV-B. The first access fetches the dedicated 99 B header buffer starting at address 24 in the sanity check phase. In the setup phase, the boundary scan register of the FPGA is configured with the pattern read from data block 2 located at address 132 (see Fig. 3). In the erase phase, the some design-related information provided in data block 4 (address 310) and 1 (address 123) is written to the user row area within the FPGA. During the write and verification phases, the requested addresses are strictly linear and increasing by the page size while iterating data block 5. Finally, data block 6 is read with two additional page requests in the release phase. This page access sequence is only influenced by the selected page size and the targeted FPGA device, but not by the actual hardware design represented by the bitstream, even if page compression is activated. The page access sequence can thus be recorded for a baseline configuration process and hard-coded in the gateway firmware for proper page prediction.

Fig. 7 shows, how the page prediction is integrated into the page loading procedure. The predicted page request sent from

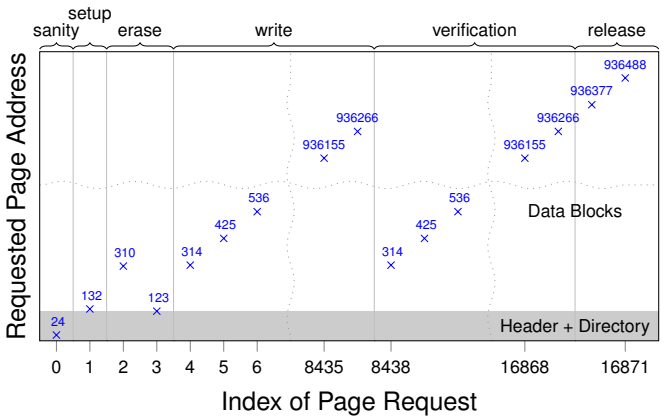


Fig. 6. DirectC page access sequence for the AGL1000 device and a page size of 111 B. Note the dotted axis discontinuities.

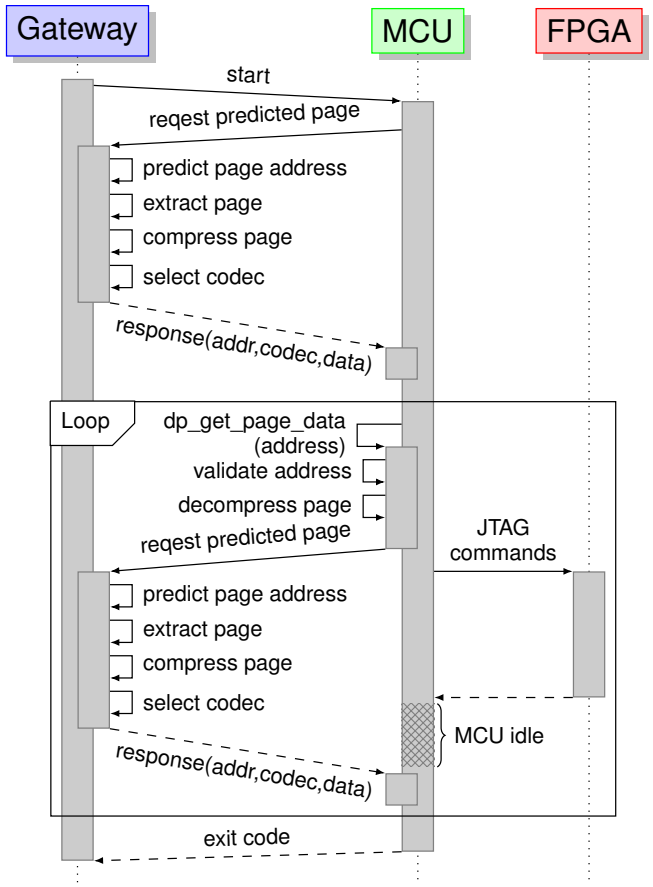


Fig. 7. OTAP page loading procedure with page prediction and compression

the MCU to the gateway does not include a page address. Instead, this address is predicted at the gateway based on the sequence of already transmitted pages, as described above. The optionally compressed page response also includes the predicted page address. The actual DirectC page load function (`dp_get_page_data`) first compares the address of the last received predicted page with the actually requested page address. Also unlikely, this validation may fail due to network communication errors, e.g., superfluous retransmissions due to missed acknowledgements. Those mispredictions are not DirectC-related, but have to be handled by falling back to unpredicted explicit page requests. After a positive address validation, the received page is decompressed and the next predicted page is requested. Instead of waiting for the response, DirectC can immediately continue to process the last received page. In the best case, the next page response is received before the next call to the page load function, thus completely eliminating the stalling of the MCU.

V. Evaluation

To evaluate the concepts proposed in Section IV, the improved DirectC paging mechanism was implemented on the HaLOEWEn mote (version 3), mainly consisting of a Microsemi AGL1000 FPGA used as hardware-accelerator and a Texas Instruments CC2531 Radio System-on-Chip (RF-SoC)

TABLE I
COMPARISON OF HARDWARE DESIGNS USED TO GENERATE BITSTREAMS

Design	Logic Cells	Utilization	Description
D1	716	3 %	Generic application: $y = x^5$
D2	5677	23 %	Generic application: $y = x^{50}$
D3	20714	84 %	Realistic SHM application [19]

to interface the WSN. The latter combines an 8 bit MCU with a IEEE 802.15.4 wireless transceiver capable of handling 128 B frames with up to 116 B payload at 250 $\frac{\text{kbit}}{\text{s}}$. The maximum DirectC page size transferable without frame segmentation is limited to 111 B, as another 4 B field for the page address and 1 B field for the selected codec have to be included in page response frames (see Fig. 7). Maximizing the page size minimizes the overhead caused by the frame header. Thus, a page size of 111 B is used throughout the remaining evaluation.

To evaluate the compressibility of the bitstreams, three different hardware-designs shown in Table I have been synthesized. While the first two are of synthetic nature used to easily scale the core utilization, the third represents a realistic design in the Structural Health Monitoring (SHM) application domain of the heterogeneous sensor node.

A. Memory Footprint

The memory required on the for compression and page prediction is not critical, as the gateway node is considered to be not resource-constrainded. Thus, only the memory footprint for the MCU on the heterogeneous sensor nodes are reported here.

When disabling all DirectC features not required for the AGL1000 core programming (e.g., without bitstream security) and compiling the firmware with the Small Device C Compiler (version 3.5.0), the resulting footprint for the three CC2531 memory regions amounts to 117 B (46 % of 256 B) internal DATA, 528 B (6 % of 8 kB) external DATA, and 19.8 kB (7.7 % of 256 kB) CODE. Nearly 40 % of the occupied external DATA is used for the page and header buffer. The page decompression is performed in-place and thus does not increase the memory footprint off the constrained MCU.

B. Compression Ratio

First, the compression ratio achievable by the different compression schemes are evaluated. Fig. 8 shows the compression ratio (i.e., the compressed size relative to uncompressed size) achieved for different combinations of compression schemes selectable for each page. The page sequence actually transmitted during the complete configuration procedure (Fig. 6) is taken into account. Thus, duplicated page transmissions are also considered.

For the first three bar groups of Fig. 8, a specific compression scheme was enforced. As expected, the compression ratio improves with the complexity of the codec and degrades with the core utilization of the bitstream. If all compression schemes (including the uncompressed transmission) can be selected for every transmitted page, the compression ratio improves even further and reaches 20 % in the best case.

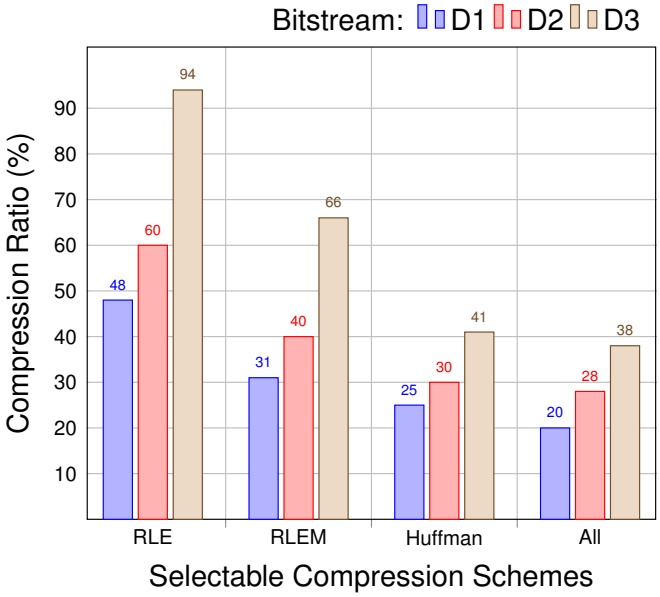


Fig. 8. Comparison of achievable compression ratios for different bitstreams.

C. Configuration Time

To evaluate the influence of the page compression and prediction mechanism on the overall configuration time, the five scenarios shown in Fig. 9 were considered. Each of the three bitstreams was configured ten times for each scenario. The observed worst case standard deviation was smaller than 0.5 s and is thus not shown in Fig. 9. All improvements described in Sections IV-A and IV-B are applied.

Without page compression and page prediction (first bar group), the overall OTAP process takes 711 s. This duration

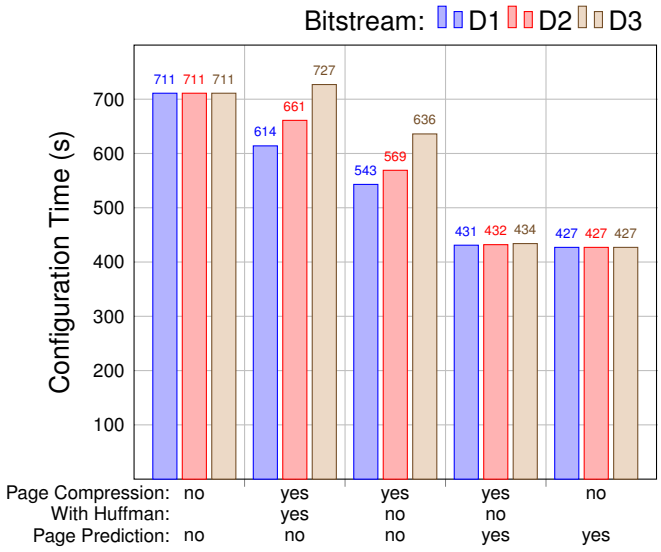


Fig. 9. Comparison of OTAP duration for different bitstreams and paging strategies.

is only slightly improved by activating the page compression (second bar group) for the simple bitstreams D1 and D2. Even worse, the execution time for D3 is actually increased. Detailed profiling revealed the Huffman decoder being the bottleneck, so the Huffman codec was completely disabled for the third bar group. With disabled Huffman codec, the page compression reduces the configuration time for the three bitstreams by up to 24 %.

When combining page compression and page prediction (fourth bar group), the configuration time is improved even further, reaching up to 39 % reduction. Surprisingly, the page prediction without page compression (fifth bar group), achieves even better results (i.e., 40 % configuration time reduction). The main reason for this finding is that the page prediction already hides the page transfer time behind the actual page evaluation, as shown in Fig. 7. Another reduction of the page transfer time, as it is achieved by the compression stage, thus does not improve the overall configuration time. Even worse, the required page decompression actually slows down the configuration process. Therefore, page prediction should be utilized *instead* of page compression. Only if page prediction has to be (temporarily) disabled due to network-related prediction erros, page compression should be activated.

To compare the configuration duration with the results achieved by the related reseach described in Section II, the deviating bitstream sizes have to be taken into acount. When normalizing the configuration time to the bitstream size, the OTAP implementation proposed in this work requires 0.47 $\frac{s}{kB}$ and thus clearly outperforms [14] with 13.6 $\frac{s}{kB}$, and still slightly beats [13] with 0.5 $\frac{s}{kB}$.

D. Configuration Energy

To evaluate the overall energy spent during a configuration process, the voltage and current at two supply rails were sampled with a Hitex PowerScale device [20]. While the 3 V rail (V_{DD}) supplies the FPGA and RF-SoC core (including the wireless transceiver), the 3.3 V rail (V_{PUMP}) is only required during the Flash erase and write phase. Fig. 10 shows the captured current flow for a configuration of bitstream D0 with enabled page compression and page prediction. The voltage and current samples were captured with the highest achievable sampling frequency of 75 kHz to record even shorter current spikes caused by the switching regulators of the HaLOEWEn power supply architecture. The overall energy was derived by integrating the current-voltage product over the configuration time. Less than 3 % of the overall configuration energy is drawn by the V_{PUMP} rail.

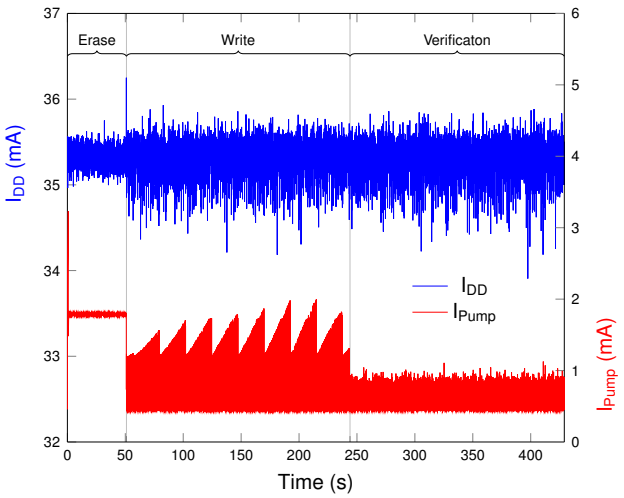


Fig. 10. Current consumed during the configuration by both supply rails: 3.3 V V_{Pump} for the IGLOO Flash modification, and 3 V V_{DD} for all other components.

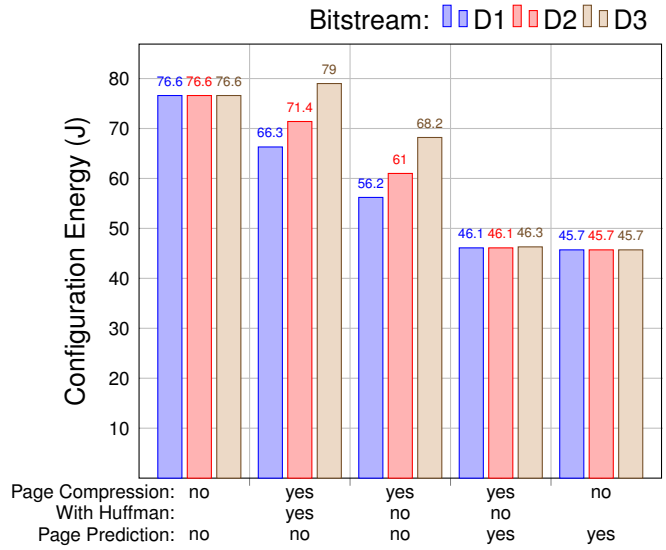


Fig. 11. Comparison of OTAP energy consumption for different bitstreams and paging strategies.

To evaluate the influence of the page compression and prediction mechanism on the overall energy consumption, the five scenarios shown in Fig. 11 were considered. When comparing Figs. 9 and 11, similar insights regarding the page compression and prediction can be derived for the energy consumption. The best results (i.e., 40 % energy savings) are achieved for the page prediction *without* page compression.

VI. Conclusion

In this paper, the improved implementation of an OTAP procedure targeting Microsemi FPGAs in heterogeneous WSN motes has been demonstrated and evaluated. The implementation is based on the DirectC library provided by Microsemi for JTAG programming. The main improvements of the DirectC paging mechanism involve the lossless compression of the transmitted pages and the prediction of page requests. The latter effectively reduces the idle time of the MCU controlling the configuration process. The page prediction was found to be the most effective improvement, as it reduces the time and energy required for the configuration by 40 % independently from the actual bitstream content.

Some future work is required to further improve the efficiency and robustness of the proposed OTAP procedure. First, the current implementation does not fully exploit the dynamic

power management capabilities of the HaLOEWEn mote. For example, the wireless transceiver is currently not turned off as long as no bitstream page is expected to be received. Second, the efficient and simultaneous reconfiguration of multiple sensor nodes with the same, or slightly different bitstreams should be investigated. Bitstream pages required by multiple nodes should be broadcast, instead of using multiple transmissions of the same page. Finally, the configuration traffic should also be protected against malicious modifications by an appropriate encryption scheme.

REFERENCES

[1] A. Engel and A. Koch, "Heterogeneous Wireless Sensor Nodes that Target the Internet of Things," *IEEE Micro*, vol. 36, no. 6, pp. 8–15, 2016.

[2] K. Shahzad and B. Oelmann, "Quantitative Evaluation of an FPGA based Wireless Vibration Monitoring System in relation to Different Sampling Rates," in *Proc. of the Int. Conf. on Circuits and Systems*, 2014.

[3] J. Valverde, A. Otero, M. Lopez, J. Portilla, E. de la Torre, and T. Riesgo, "Using SRAM Based FPGAs for Power-Aware High Performance Wireless Sensor Networks," *Sensors*, vol. 12, no. 3, pp. 2667–2692, 2012.

[4] A. Brokalakis, G.-G. Mplemenos, K. Papadopoulos, and I. Papaefstathiou, "RESENSE: An Innovative, Reconfigurable, Powerful and Energy Efficient WSN Node," in *IEEE Int. Conf. on Communications*, 2011, pp. 1–5.

[5] G.-G. Mplemenos and I. Papaefstathiou, "Fast and power-efficient hardware implementation of a routing scheme for WSNs," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2012, pp. 1710–1714.

[6] F. Philipp and M. Glesner, "An event-based middleware for the remote management of runtime hardware reconfiguration," in *23rd Int. Conf. on Field programmable Logic and Applications*, 2013, pp. 1–4.

[7] O. Berder and O. Sentieys, "PowWow : Power Optimized Hardware/Software Framework for Wireless Motes," in *23rd Int. Conf. on Architecture of Computing Systems (ARCS)*, 2010, pp. 1–5.

[8] V. Rosello, J. Portilla, and T. Riesgo, "Ultra low power FPGA-based architecture for Wake-up Radio in Wireless Sensor Networks," in *37th IEEE Annual Conf. on Industrial Electronics Society (IECON)*, 2011, pp. 3826–3831.

[9] Microsemi. DirectC. [Online]. Available: https://www.microsemi.com/products/fpga-soc/design-resources/programming/directc

[10] A. Biedermann, B. Dreyer, and S. Huss, "A generic, scalable reconfiguration infrastructure for sensor networks functionality adaption," in *IEEE Int. SOC Conference*, 2013, pp. 301–306.

[11] G. D. Mois, M. Hulea, S. Folea, and L. Miclea, "Self-healing capabilities through wireless reconfiguration of FPGAs," in *9th East-West Design Test Symposium (EWDTS)*, 2011, pp. 22–27.

[12] P. Ruberg, A. Guitar, and P. Ellervee, "Flexible controller for educational robot kit," in *2015 IEEE Int. Conf. on Microelectronics Systems Education (MSE)*, 2015, pp. 17–20.

[13] I. Adly, H. F. Ragai, A. El-Hennawy, and K. A. Shehata, "Over-The-Air Programming of PSoC sensor interface in wireless sensor networks," in *15th IEEE Mediterranean Electrotechnical Conference (Melecon)*, 2010, pp. 997–1002.

[14] Y. E. Krasteva, J. Portilla, E. de la Torre, and T. Riesgo, "Embedded Runtime Reconfigurable Nodes for Wireless Sensor Networks Applications," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1800–1810, 2011.

[15] S. Yamaguchi, T. Miyazaki, J. Kitamichi, S. Guo, T. Tsukahara, and T. Hayashi, "Programmable wireless sensor node featuring low-power FPGA and microcontroller," in *Int. Joint Conf. on Awareness Science and Technology and Ubi-Media Computing*, 2013, pp. 596–601.

[16] Y. Wee and T. Kim, "A new code compression method for FOTA," *IEEE Trans. on Consumer Electronics*, vol. 56, no. 4, pp. 2350–2354, 2010.

[17] Microsemi. STAPL Player. [Online]. Available: https://www.microsemi.com/products/fpga-soc/design-resources/programming/stapl-player

[18] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[19] A. Engel, T. Siebel, and A. Koch, "A Heterogeneous System Architecture for Low-Power Wireless Sensor Nodes in Compute-intensive Distributed Applications," in *40th IEEE Conf. on Local Computer Networks*, 2015.

[20] Hitex. PowerScale with ACM technology. [Online]. Available: https://www.hitex.com/de/tools/energy-optimization/powerscale