IEICE TRANS. ELECTRON., VOL.E94–C, NO.7 JULY 2011

PAPER

An Energy Efficient Sensor Network Processor with Latency-Aware Adaptive Compression*

Yongpan LIU[†a)], ***Member***, **Shuangchen LI**[†], **Jue WANG**[†], **Beihua YING**[†], ***and*** **Huazhong YANG**[†], ***Nonmembers***

SUMMARY This paper proposed a novel platform for sensor nodes to resolve the energy and latency challenges. It consists of a processor, an adaptive compressing module and several compression accelerators. We completed the proposed chip in a 0.18 μm HJTC CMOS technology. Compared to the software-based solution, the hardware-assisted compression reduces over 98% energy and 212% latency. Besides, we balanced the energy and latency metric using an adaptive module. According to the scheduling algorithm, the module tunes the state of the compression accelerator, as well as the sampling frequency of the online sensor. For example, given a 9 μs constraint for a 1-byte operation, it reduces 34% latency while the energy overheads are less than 5%.

key words: *wireless sensor network, compression accelerator, energy efficient, latency aware, adaptive compression*

1. Introduction

Wireless Sensor Networks (WSNs) provide a convenient way to collect the distributed information. They help humans to know the physical world better. Sensor networks contain many nodes powered by the small-size batteries. They have strict limits on the volume and the cost of the nodes. Thus, the power consumption should be minimized to satisfy the yearlong lifetime need.

Previous results [2] showed the energy to wireless transfer 1-bit data is about several magnitudes larger than that of a 32-bit calculation. And the nearby readings of sensors are likely correlated. Therefore, we can compress the original data before the transmission, which significantly reduces the wireless transferred data. Various compressing techniques have been used in structural health recording [3], body sensor network [4], video surveillance [5] and environmental monitoring [6].

In real deployments, low-performance processors (i.e. ATmel128, MSP430, etc.) are used on a low cost platform, such as MicaZ, TelosB. Those processors are good at control but inefficient at compression. In [7], the processor consumed over 64% power to run a simple application. Thus, the hardware-assisted compression is promising to reduce the energy and put the processor into sleep.

Besides, the compression technique depends on input, hardware and accuracy. Researchers had been aware of energy and latency overheads of the compression. References [2], [8] pointed out that we should be carefully to use a complex compressing algorithm to save energy. Ying et al. [9] showed the software-based compression is not always energy efficient in a sensor node. They set up an arbitration system to avoid unnecessary compressions for energy savings.

Recently, the real-time applications put various needs on the latency of WSNs. They include target tracking, track recording, structural health diagnosis, etc. Li et al. [10] studied the present real-time solutions for WSNs. They consist of MAC and routing protocols, data processing strategies and cross-layer designs. In the detecting applications [11], [12], the latency above a threshold can lead to an eventual failure. Therefore, the latency metric becomes mandatory in the real-time WSNs. To our best knowledge, none of existed work had explored the latency savings using the hardware-assisted compression.

The contributions of this article are listed as below. We proposed an energy efficient architecture for sensor nodes. It includes a processor, an adaptive module, a power-efficient interconnection and several compression accelerators (CA). We provided a thorough analysis on the hardware aided compression under energy and latency metrics. Furthermore, we completed the proposed architecture in real silicon. Compared with the pure software approach, measured results showed over 98% energy and 212% latency savings.

In addition, we proposed an adaptive framework for the chip. Compared to the static system with a high false ratio, the users can smoothly adjust the tradeoff between latency and energy. Given a 9 μs constraint for a 1-byte data operation, the framework can reduce 34% latency, while the energy overheads are less than 5%.

We organize the rest of this paper as follows. Section 2 explains the motivation and related work in this area. In Sect. 3, we describe the node architecture, the online compression arbitration, and the scheduling algorithm. Section 4 describes the overall architecture, the CA unit and the online sensor. We present the results of the chip and the compression arbitration in Sect. 5. Finally, Sect. 6 concludes the paper and gives out the future work.

2. Related Work

There are already some works on the hardware aided compression. Reference [13] proposed a hardwired compression unit, which reduced the traffic in the cache-to-memory path. However, they verified the design by the architecture-level

Manuscript received August 30, 2010.
Manuscript revised February 15, 2011.

[†]The authors are with the Electronic Engineering Department, Tsinghua University, 100084, Beijing, P.R. China.

*This paper was presented at GLSVLSI2009 [1].

a) E-mail: ypliu@tsinghua.edu.cn

DOI: 10.1587/transele.E94.C.1220

simulation. Kim et al. [4] designed a processor for the body sensor network. It contains a coprocessor for the biological compression, but a new compiler is needed. Hempstead et al. [14] presented an accelerator-based processor. They validated the accelerator is power efficient for the network operations. Recently, Seok et al. [15] realized a Huffman encoder for the data compression in Phoenix processor. However, no one has discussed the adaptive compression. This paper proposed an architecture to support this feature.

The architecture also needs a low power interconnection. There are plenty of bus specifications, such as AMBA [16], Avalon [17], Wishbone [18], GALS [19], etc. However, they are too complex for the WSN applications. The communicating protocol should be both simple and power efficient. Therefore, we proposed a scalable interconnection for the multiple CAs.

Plenty of compressing algorithms have been used in WSNs to save energy. Mainwaring et al. [6] set up a habitat monitoring WSN for seabirds. They investigated Delta, Huffman coding and Lempel-Ziv algorithms. Chiasserini et al. [5] realized a low-cost JPEG algorithm for video surveillance. They concluded the complex algorithms will not save energy in WSNs. Structural recording [3] explored the wavelet-based compression to overcome the narrow bandwidth imposed by the low-power wireless radios. However, the Mica2 platform only realized the run-length encoding.

All of above realized compression algorithms in software. Phoenix processor contained a hardware aided compression module. However, a thorough analysis on the adaptive CA is unavailable. Furthermore, the compression algorithms are sensitive to input and platform. They greatly impact the balance between compression and transmission. Our adaptive arbitration can overcome those challenges and suitable for other hardware aided compression chips.

3. Chip Architecture

This section first discusses the diagram of the sensor node. Furthermore, we illustrate the low-power chip architecture and the communication protocols. Finally, we show the compression arbitration and the scheduling algorithm.

3.1 System Diagram

In Fig. 1, the processor compresses the readings and sends them via the transceiver in the traditional flow (black arrow line). The proposed architecture provides an energy-efficient CA (dotted line with arrow). The hardware aided compression consumes magnitudes less power but with reasonable silicon overheads.

3.2 Chip Structure

Figure 2 shows the details of the MCU and CA blocks in Fig. 1. The architecture consists of a general-purpose processor, controllers, CAs and a memory module. The processor is connected to the I^2C, SPI and $UART$ controllers

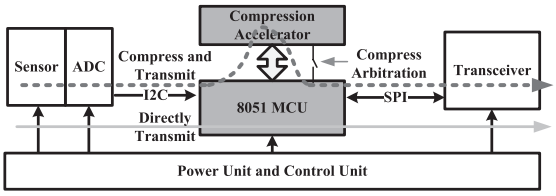

Fig. 1 Illustration of WSN compression.

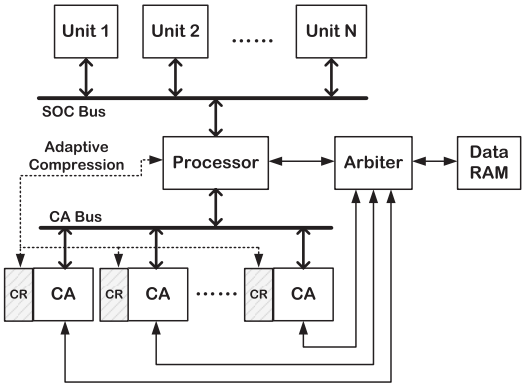

Fig. 2 Proposed architecture of a sensor node.

by the SoC bus. And the processor can configure each compression accelerator through the CA bus. The data transfer between processor and CA is using SRAM, coordinated by the Arbiter. Each CA has an online sensor for the compression ratio (CR). It provides the real-time CR to the processor. Based on the CR, the scheduling algorithm realizes the adaptive compression.

The proposed architecture supports different CAs to process the varying data. The processor will turn on one CA, after it turns off others to eliminate static power. Once the processor finished the configuration, it carries out a task or goes into sleep.

Both the processor and the CAs can access the SRAM to avoid the frequent data transfers. The processor can select the CA unit and signal it by giving the target SRAM address to the register. The CA unit will update the status register when it finished the compression.

3.3 Communication Protocol

In WSNs, the proposed architecture need support different CAs to compress various data in parallel. This paragraph explains the protocol of the CA bus. It supports a tunable number of the CAs. Figure 3 shows the communication process, and we describe the protocol as below.

1. The processor decides the CA identification and indicates it through the signal ***Sel***. The CA number determines the width of ***Sel***.
2. The processor provides identification, address, and other parameters to the arbiter before compression.
3. The processor sets the ***Ctrl*** line and signals the CA to work.

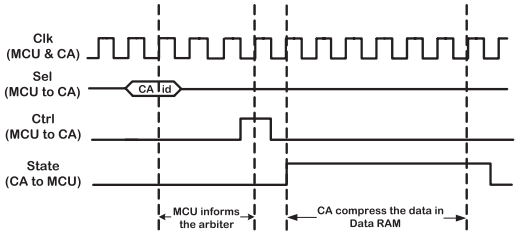

Fig. 3 Protocol of CA bus.

4. The CA unit sets the ***State*** line and compresses the target data in the SRAM. Simultaneously, the processor can execute other tasks or go into sleep.
5. The CA unit pulls down the ***State*** line to tell the end of the compression. The CA bus returns to the idle state.

The arbiter coordinates the data operations based on the protocol. In the read mode, it generates the target address and fetches the data from the SRAM. In the write mode, it transfers data to SRAM from the processor or the CAs. The arbiter chooses the address signal from the processor in the normal mode. It selects the address from the CAs in the compression mode.

3.4 Adaptive Compression Diagram

Adaptive compression need balance the computing overheads and the transmission savings. We first provide the equations to compute the energy consumption in the direct and indirect transmission mode as the followings:

$$E_{uncomp} = P_{RF} * T_{RF} * L \quad (1)$$

$$E_{MCUcomp} = (P_{MCU} * T_{MCU} + P_{RF} * T_{RF} * CR) * L \quad (2)$$

$$E_{CAcomp} = (P_{CA} * T_{CA} + P_{RF} * T_{RF} * CR) * L \quad (3)$$

- where E_{uncomp}, $E_{MCUcomp}$ and E_CAcomp separately denotes the energy consumption under the direct transmission mode, the MCU compressed and transmission mode and the CA compressed and transmission mode,
- P_{RF}, P_{MCU} and P_{CA} are the power consumption of the RF transceiver, the processor and the CA,
- T_{RF}, T_{MCU} and T_{CA} are the wireless transferring time, the compressing time of the processor and the CA per byte data,
- L is the original data size, and
- CR is the compression ratio defined as the compressed data size over the original data size.

In the latency-sensitive applications, we can reduce the latency by compression. The CR determines the extent of the reduction. The following equations show the latency in the direct and indirect transmission mode:

$$LA_{uncomp} = L * T_{RF} \quad (4)$$

$$LA_{MCUcomp} = (T_{MCU} + T_{RF} * CR) * L \quad (5)$$

$$LA_{CAcomp} = (T_{CA} + T_{RF} * CR) * L \quad (6)$$

where LA_{uncomp}, $LA_{MCUcomp}$ and LA_{CAcomp} is the latency of transmitting directly and transmitting after the processor or CA compression.

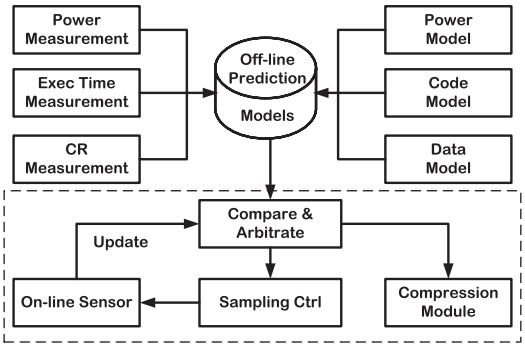

Fig. 4 Adaptive compression flow.

Figure 4 shows the adaptive compressing mechanism. The off-line prediction models are built based on the power, code and data models. Their parameters are extracted from the real measurements. Equations (1)–(3) define the power models. The data model denotes the trend of the CR variations along the time. The code model adopts the instruction-level model. The total instruction number of the code is extracted after compiling. The execution time of the code is obtained by accumulating the cycle number of each instruction. The online sensor provides the real-time CR to the comparison and arbitration module. The module tunes the sampling rate of the sensor, as well as the enable signal of the compressing module. The scheduling algorithm in Sect. 3.5 ensures the convergence of the sampling adjustment.

3.5 Scheduling Algorithm

In the adaptive compression, we compare the real-time CR with the reference CR_0. As Sect. 4.2 shown, an online sensor updates the CR periodically. The optimizing objective determines the reference CR_0. We give out the definition of the CR_0 under three cases: minimal energy, minimal latency and latency-constrained minimal energy.

1. Minimal Energy According to Eqs. (1) and (3), the reference compression ratio CR_E is expressed as below, when the energy consumption of the computation equals to that of the communication.

$$CR_E = \begin{cases} 1 - (P_{MCU} * T_{MCU})/(P_{RF} * T_{RF}) & \text{if MCU} \\ 1 - (P_{CA} * T_{CA})/(P_{RF} * T_{RF}) & \text{if CA} \end{cases} \quad (7)$$

When $CR < CR_E$, the system can reduce the energy consumption. Thus, the CR_0 equals to the CR_E.

2. Minimal Latency According to Eqs. (4) and (6), the balanced compression ratio CR_L can be expressed as below, when the latency of the transmission equals to that of the computation.

$$CR_L = \begin{cases} 1 - T_{MCU}/T_{RF} & \text{if MCU} \\ 1 - T_{CA}/T_{RF} & \text{if CA} \end{cases} \quad (8)$$

Algorithm 1 Adaptive Compression Tuning Algorithm

Input: CR, $PreState$, $CurInter$
Output: $NextState$, $NextInter$
1: $PreState = FALSE$, $CurInter = RMax$, $Step = S0$
2: **if** (*Next Arbitration Request Ready*) **then**
3: **if** $(CR > CR_0)$ & $(PreState = TRUE) \| (CR < CR_0)$ & $(PreState = FALSE)$ **then**
4: **if** $CurInter == RMin$ **then**
5: $NextState = NOT(PreState)$
6: **else**
7: $NextInter- = Step$
8: $NextState = PreState$
9: **end if**
10: **else**
11: **if** $(CurInter + Step) \leqslant RMax$ **then**
12: $NextInter+ = Step$
13: **end if**
14: $NextState = PreState$
15: **end if**
16: **end if**

When $CR < CR_L$, we reduce the transferring latency. Thus, the CR_0 equals to the CR_L.

3. Latency-Constrained Minimal Energy In this case, we need minimize the energy consumption under a specific constraint of the latency. The corresponding reference CR_0 can be expressed by the following equation:

$$CR_{LA} = \begin{cases} (max(\frac{LA}{L}, T_{RF}) - T_{MCU})/T_{RF} \\ \quad \text{if } CR_E > CR_L \text{ MCU} \\ (max(\frac{LA}{L}, T_{RF}) - T_{CA})/T_{RF} \\ \quad \text{if } CR_E > CR_L \text{ CA} \end{cases} \tag{9}$$

$$CR_0 = min(CR_E, CR_{LA}) \tag{10}$$

where LA denotes the specific latency. Next, we design a scheduling algorithm to tune the operating state of the compression module, as well as the sampling rate of the sensor.

The inputs are the real-time CR, the previous compression status $PreState$ and the current sampling interval $CurInter$. The outputs are the next compression status $NextState$ and the next sampling interval $NextInter$. In the initialization stage, the compression status is set as $False$. The current sampling interval equals to the maximal value $RMax$ and the tuning step $Step$ equals to $S0$. When the arbitration request is ready, the conditions in line 3 are tested. If unnecessary compression or potential energy or latency savings are detected, line 4–9 decrease the sampling interval. When the sampling interval reaches the minimal value $RMin$, we assign the inverse value of $PreState$ to $NextState$. Otherwise, the arbitration module increases the sampling interval until the maximal value $RMax$ is reached.

4. Chip Implementation

This section demonstrates the implementation of the proposed architecture for a specific application. We describe the architecture in Sect. 4.1. Section 4.2 explains the selected compression algorithm, the CA and the online sensor.

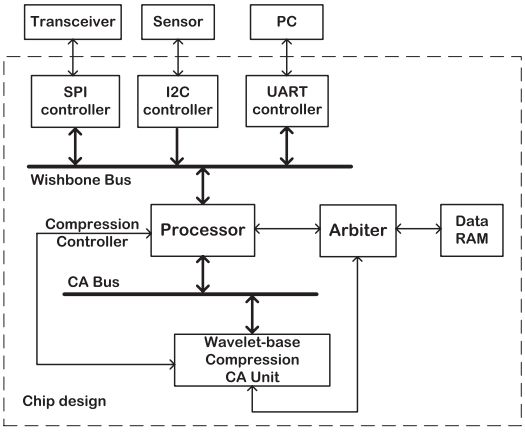

Fig. 5 The SoC architecture of the proposed chip.

4.1 Node Processor

To demonstrate the efficiency of the architecture, we implemented a sensor network processor containing a general processor, one CA unit and several interface controllers. Figure 5 shows the diagram of the platform, whose components are illustrated as the followings:

1. Processor: we adopted a MC8051-compatible processor using the Keil C compiler.
2. *SPI* controller: it connects the Wishbone bus to the external transceiver. Most of available transceivers, such as CC2420, ZL70250, etc., support the SPI interface.
3. I^2C controller: it connects the Wishbone bus to the sensor to acquire the sampling data.
4. *UART* controller: it connects the Wishbone bus to a PC to show the debugging information.
5. CA unit: it executes the hardware aided compression in Sect. 4.2.
6. Wishbone bus: it connects the processor to the interface controllers.

Although one CA unit is implemented in this chip, the major difference between a single CA chip and a multiple CA chip is the setting of the signal ***Sel*** before the compression.

4.2 Compression Accelerator

Due to its advantages to balance the power distribution, we choose a distributed compressing algorithm for the sensor node. Among the existed algorithms, such as LTC [20], MCS [21], TREG [22], DPCM [23], they are either too complex for the hardware implementation or need impractical pre-knowledge of the total network. Therefore, we choose the distributed wavelet compression [24] with a light workload as the candidate[†]. The selected algorithm can process spatially and temporally distributed data. It corresponds to

[†]It should be noted the SoC architecture is not limited to the wavelet-based compression. It also supports other compression algorithms in other scenarios.

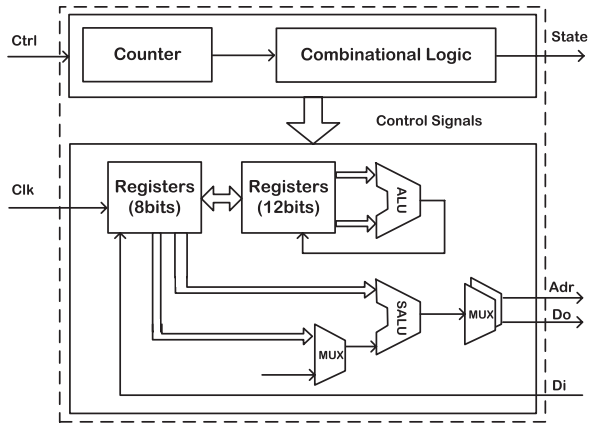

Fig. 6 Architecture of the compression accelerator.

the data from several nodes or from a single node in different time slots. The algorithm consists of two steps as the followings:

1. Wavelet Transformation

In the distributed wavelet compression, the sensor nodes are split into odd and even subsets. Each node calculates high-pass and low-pass coefficients through the wavelet transformation. The biorthogonal wavelets are used based on the lifting scheme. The formulas to compute the 5/3 wavelet coefficients are shown:

$$d(2n+1) = D(2n+1) - \frac{1}{2}[D(2n) + D(2n+2)] \quad (11)$$

$$s(2n) = D(2n) + \frac{1}{4}[d(2n-1) + d(2n+1)] \quad (12)$$

where $d(n)$ and $s(n)$ denotes high-pass and low-pass coefficients of the node n; $D(n)$ represents the original readings from the node n. Partial coefficients are used to guarantee all sensor nodes to communicate in one direction [24]. If a 2-level wavelet compression is adopted, we further split the even sensor nodes into odd and even subsets and do the wavelet transformation to the low-pass coefficients from the 1-level transformation.

2. High-pass Coefficients Encoding

We encode low-pass and high-pass coefficients from the wavelet transformation. The high-pass coefficients are quite small due to the strong correlation among the nodes. We can use fewer bits to represent them and the run length encoding (RLE) is adopted.

Figure 6 shows the architecture of the CA. It consists of two units: a hardwired controller and a specific datapath. The hardwired controller is a state machine, including a counter and a combinational circuit. The specific datapath contains two registers, several multiplexors and specific arithmetic logical units (ALU) for the wavelet compression. The ALUs realize the special operations of the wavelet compression, such as the comparison operation during the encoding stage. The signal ***Clk*** is the system clock. According to the protocol, the signal ***Ctrl*** gets the control command from the processor, and the signal ***State*** indicates the completion of the CA. The signals ***Adr***, ***Do***, ***Di*** are used to communicate with the arbiter. The signal ***Adr*** transfers the address between the accelerator and the RAM. The signals ***Do*** and ***Di*** read and write data between the accelerator and the RAM.

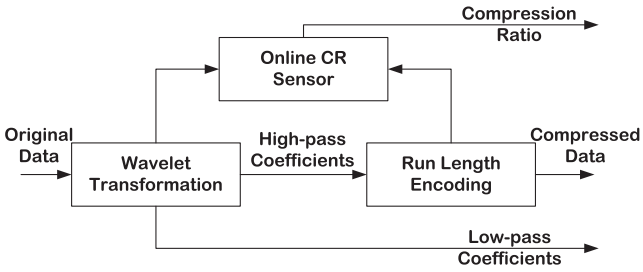

Fig. 7 Architecture of online sensor.

In order to detect the CR, we can calculate the CR from the processor or the dedicate sensor. In this implementation, we use the processor to compute the CR and send it to the scheduling algorithm. However, a dedicated sensor will be integrated in the future design. We show the diagram of a dedicated sensor in Fig. 7. The sensor counters the byte number of the compressed data and outputs the CR to the processor.

5. Experimental Results

This section first illustrates the experimental configuration. Furthermore, the hardware-assisted compression is compared with the software compression. Finally, the adaptive compression is demonstrated.

5.1 Experimental Setup

We realized the proposed chip for sensor applications by the 1-poly 6-metal HJTC 0.18 μm CMOS technology. Figure 8 shows the dimensions of the overall chip are 1.5 mm $\times$ 3 mm. The volume of the code memory, the internal data memory and the external data memory are 32 K, 256 and 8 K byte. The CA contributes to 5.2% of the total core area. We developed a prototype board in Fig. 9 to test the chip. The chip operates at a supply voltage of 1.8 V and a clock frequency of 10 MHz. The program is stored in the external EEPROM. The power consumption of the chip is measured by a data acquisition device (DAQ) from National Instruments. The DAQ is attached to the USB port of a notebook running LabView. The inputs to the CA use the real data from the moored ocean buoys project [25], including temperature, humidity, etc.

5.2 Hardware vs. Software Compression

We measured the power consumption of the CA under different inputs. The average value is 0.98 mW, whereas the average power consumption for the processor to execute the same compression is 2.61 mW. It indicates that the CA can save over 62% power consumption compared with the software solution. Figures 10 and 11 shows the measured re-

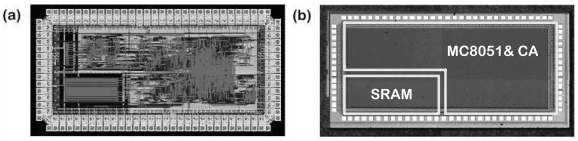

Fig. 8 (a) Chip layout (b) Chip die photo.

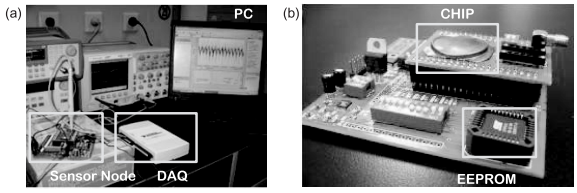

Fig. 9 (a) Chip Test environment (b) Chip demonstration board.

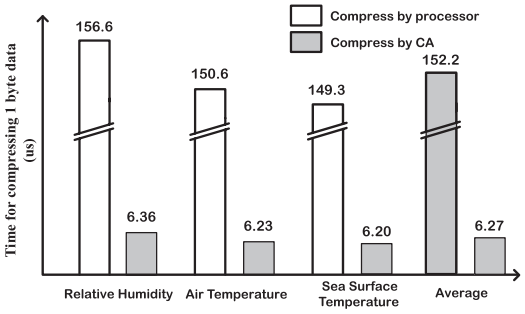

Fig. 10 Comparison of execution time.

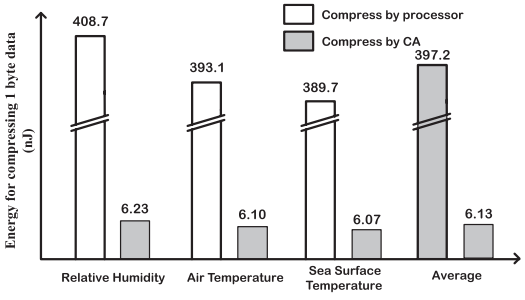

Fig. 11 Comparison of energy consumption.

sults of time and energy to compress 1-byte data of different types. It demonstrates that the CA reduces 96% execution time and 98% energy consumption compared with the processor-based approach. Though the execution time and the energy consumption may vary under different inputs, the maximal fluctuation range is less than 3%. It implies that the proposed CA can effectively reduce both latency and energy for those real data sets.

We show the energy savings of a sensor node, containing the fabricated chip and a wireless transceiver. As the *CR* may affect the energy consumption, the *CR* is set to the average measured value 50%. The adaptive experiments will provide the real-time energy consumption under various *CRs*. Table 1 shows the energy consumption of the sensor node to receive or transmit 1-byte data without or with the compression. The energy consumption using different transceivers is also given. We define the baseline as the energy consumption without the compression in the second line. The energy consumption with the MCU-based and the CA-based compression is denoted in the third and fourth lines. The MCU-based compression can save energy by 31% with the transceivers, such as CC2420 and TR1000. However, it failed in the node with a ultra low-power transceiver(ZL70250). It transfers data more energy efficient than the compressed and send operation. But the CA-based compression can always save energy with different wireless transceivers. The maximal savings are close to 50%.

Table 1 Energy of a sensor node to process 1 byte data.

OP. Mode	CC2420		TR1000		ZL70250	
	Ene. (nJ)	Sav. (%)	Ene. (nJ)	Sav. (%)	Ene. (nJ)	Sav. (%)
W.O. Comp.	2100	0	1250	0	206	0
W. MCU Comp.	1450	31	1025	18	503	−144
W. CA Comp.	1081	49	631	50	109	47

Table 2 Latency of a sensor node to process 1 byte data.

OP. Mode	CC2420		TR1000		ZL70250	
	Lat. (μs)	Sav. (%)	Lat. (μs)	Sav. (%)	Lat. (μs)	Sav. (%)
W.O. Comp.	32	0	69	0	43	0
W. MCU Comp.	168	−425	187	−171	174	−305
W. CA Comp.	22	31	41	41	28	35

We compare the latency savings of a sensor node under different operation modes in Table 2. It lists the latency to send 1-byte data without compression, with the MCU-based and the CA-based compression in line 2–4. It is surprising that the MCU-based compression increases the latency in all cases, though it reduces the energy consumption. The maximal overhead approaches to 425%. The long computing latency by the low-performance processor leads to such results. However, the CA-based compression reduces 41% latency. Therefore, the CA-based compression significantly reduces both energy and latency.

5.3 Adaptive Compression

Figure 12 shows the energy to process and transmit 1-byte data under different static *CRs*. The MCU-based compression does not save the energy when the *CR* is bigger than the threshold CR_E. The adaptive compression recognizes this threshold and avoid unnecessary operations. Compared with the always compressing solution, the adaptive MCU-based compression brings up to 19% or 191% energy savings with CC2420 or ZL70250. The CA-based compression is energy-efficient in a large range. The typical CR_E is larger than 97%, which means the system tends to always compress unless $CR > CR_E$. The energy savings are 0.3%(CC2420) and 3%(ZL70250). Therefore, the adaptive compression leads to more energy savings when a more energy-efficient transceiver is used.

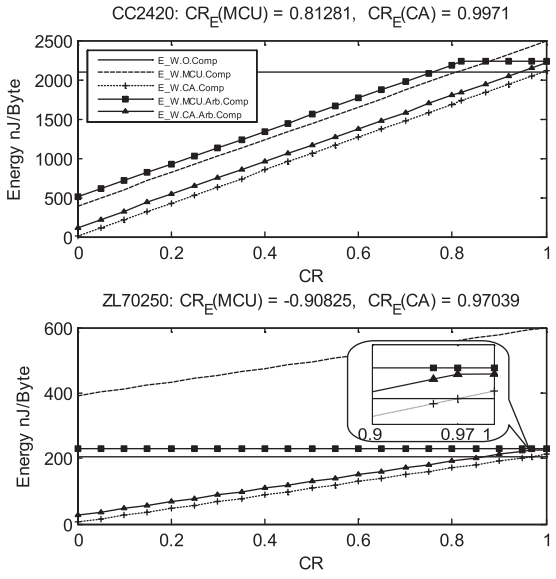

Fig. 12 Energy optimal adaptive compression.

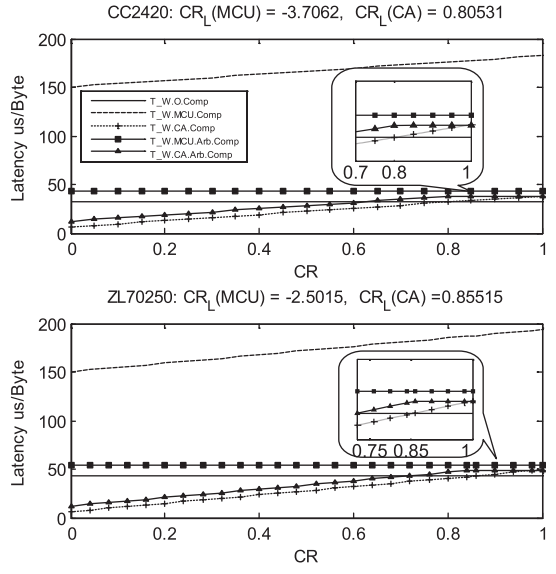

Fig. 13 Latency optimal adaptive compression.

Figure 13 shows the latency to process and transmit 1-byte data under different static *CRs*. The MCU-based compression fails in all cases. However, the adaptive compression can avoid the penalties when the latency is more critical than the energy. Compared with the always compressing solution, the latency savings of the adaptive compression is up to 471%(CC2420) and 350%(ZL70250). On the the hand, the CA-based compression brings the latency savings when $CR < CR_L$. What's more, the CA-based adaptive compression can bring up to 19%(CC2420) and 14%(ZL70250) latency savings over the always compressing solution. It indicates that the adaptive compression leads to more latency savings with a high data rate transceiver.

Figures 12 and 13 show that the optimal latency solution is inconsistent with the optimal energy one. For example, the MCU-based compression with CC2420 can save energy but increase the latency. Therefore, the latency-constrained energy optimal problem is relevant. We evaluate the latency-constrained energy optimal compression under several dynamic *CR* cases. We adopt a 2 Mbps nRF2401L transceiver with its parameters in datasheet. We denote the energy and the time overheads of the arbitration as E_{arb} and T_{arb}. The arbitration mainly consists of data sampling and comparing. The data sampling is finished by the online sensor, which can be executed in parallel. The comparing operation is a time-efficient operation. Divided by the sampling frequency, E_{arb} equals to 0.1 nJ/Byte, while T_{arb} is trivial. The experimental parameters are listed in Table 3.

Table 3 Parameters in the dynamic CR experiments.

Variable	Value	Variable	Value	Variable	Value
Time	1 (s)	L	250 (KB)	E_{arb}	0.1 (nJ/B)
R_{min}	3 (ms)	R_{max}	7 (ms)	Step	1 (ms)
E_{RF}	151.8 (nJ)	T_{RF}	4 (μs)	T_{arb}	0 (μs)

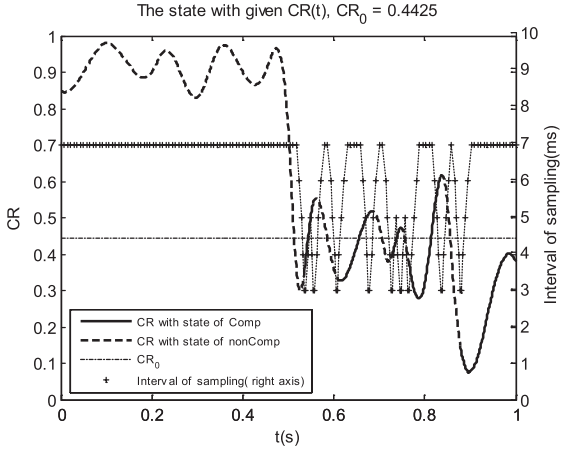

Fig. 14 Latency constrained energy optimal adaptive compression under dynamic CRs.

Figure 14 demonstrates the dynamic *CR* and sampling interval curves. Setting the latency constraint as 8 μs, the corresponding CR_0 is calculated by Eq. (9). The proposed adaptive algorithm quickly follows the vibrating *CR* curve. Once the *CR* is larger(smaller) than CR_0 for a constant time, the sampling interval increases to the maximal value R_{max} to reduce the overheads of the arbitration. We also calculate the energy and the latency values under different constraints (7-11 μs) in Table 4. We denote the direct transmission as W.O. Comp. The CA-based constant compression is represented as W. CA. The adaptive CA-based compression is denoted as W. CA Arbitration. Table 4 shows that the adaptive compression provides a smooth tradeoff. For example, we reduce 34% latency while the energy overheads are less than 5%. It is clear that the jointly consideration of latency and energy in real applications is important.

6. Conclusion

This paper proposed a novel architecture of a sensor node,

Table 4 Latency and energy comparison of a sensor node under different configurations.

Item Name		W.O. Comp	W. CA	W. CA Arbitration/Lat. Const.(μs) 7	8	9	11
Total Energy	Val.(mJ)	38	26.1	36.2	30.8	27.4	26.1
	Sav.(%)	−46	0	−39	−18	−5	−0.3
Total Latency	Val.(s)	1	2.2	1.1	1.3	1.5	2.1
	Sav.(%)	54	0	52	41	34	5
Energy/Byte (nJ)		152	104	145	123	109	105
Latency/Byte (μs)		4	8.8	4.2	5.1	5.8	8.3

including a general-purpose processor, an adaptive compressing module and a tunable number of the CAs. We fabricated the sensor node by a 1-poly 6-metal 0.18 μm HJTC CMOS technology to verify its effectiveness. A distributed wavelet compressing CA is implemented. The chip operates at a supply voltage of the 1.8 V and a clock frequency of 10 MHz. Measured results show that the hardware aided compression reduces 62% power and 98% energy consumption compared with the software solution. Furthermore, it achieves up to 41% latency savings compared with over 171% latency increases from the processor-based solution. Finally, it is relevant to jointly consider both latency and energy with compression. The adaptive compressing framework can achieve a proper tradeoff. Our future work focuses on the implementation of the arbitration module with multi-CAs and the comparison study on different compression algorithms.

Acknowledgments

The authors would like to thank Yiqun Wang, for his helpful discussions and suggestions to improve the paper. This work was supported in part by the NSFC under grant 60976032 and 6102100, Important National Science and Technology Specific Project under contract 2010ZX03006-003-01, and High-Tech Research and Development (863) Program under contract 2009AA01Z130.

References

[1] J. Wang, B.H. Ying, Y.P. Liu, H.Z. Yang, and H. Wang, "Energy efficient architecture of sensor network node based on compression accelerator," Proc. 17th Great Lakes Symposia on VLSI, pp.117–120, 2009.
[2] K. Barr and K. Asanović, "Energy-aware lossless data compression," ACM Trans. Comput. Syst. (TOCS), vol.24, no.3, p.291, 2006.
[3] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," Proc. 2nd International Conference on Embedded Networked Sensor Systems, pp.13–24, 2004.
[4] H. Kim, S. Choi, and H. Yoo, "A low power compression processor for body sensor network," 4th International Workshop on Wearable and Implantable Body Sensor Networks, pp.65–69, 2007.
[5] C. Chiasserini and E. Magli, "Energy consumption and image quality in wireless video-surveillance networks," Proc. 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp.2357–2360, 2002.
[6] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications, pp.88–97, 2002.
[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," ACM Sigplan Notices, vol.35, no.11, p.104, 2000.
[8] C. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," Proc. 4th International Conference on Embedded Networked Sensor Systems, pp.278–290, 2006.
[9] B.H. Ying, W. Liu, Y.P. Liu, H.Z. Yang, and H. Wang, "Energy-efficient node-level compression arbitration for wireless sensor networks," Proc. 11th International Conference on Advanced Communication Technology, pp.564–568, 2009.
[10] Y. Li, C. Chen, Y. Song, and Z. Wang, "Real-time QoS support in wireless sensor networks: A survey," Proc. 7th IFAC International Conference on Fieldbuses Networks in Industrial Embedded Systems, 2007.
[11] V. Gungor and O. Akan, "Delay aware reliable transport in wireless sensor networks," International Journal of Communication Systems, vol.20, no.10, pp.1155–1177, 2007.
[12] H. Li, P. Shenoy, and K. Ramamritham, "Scheduling messages with deadlines in multi-hop real-time sensor networks," Proc. 11th IEEE Real Time and Embedded Technology and Applications Symposium, pp.415–425, IEEE, 2005.
[13] L. Benini, D. Bruni, A. Macii, and E. Macii, "Hardware-assisted data compression for energy minimization insystems with embedded processors," Proc. Design, Automation and Test in Europe Conference and Exhibition, pp.449–453, 2002.
[14] M. Hempstead, G. Wei, and D. Brooks, "An accelerator-based wireless sensor network processor in 130 nm CMOS," Proc. 8th International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, pp.215–222, 2009.
[15] M. Seok, S. Hanson, Y. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw, "The Phoenix Processor: A 30 pW platform for sensor applications," Proc. 2008 IEEE Symposium on VLSI Circuits, pp.188–189, IEEE, 2008.
[16] "Amba axi protocol v1.0 specification," ARM Limited, 2004.
[17] "Avalon interface specification," Altera Corporation, 2005.
[18] "Wishbone system-on-chip (soc) interconnection architecture for portable ip cores," OpenCores.org, 2002.
[19] C. Fernández, R. Raval, and C. Bleakley, "GALS SoC interconnect bus for wireless sensor network processor platforms," Proc. 17th ACM Great Lakes Symposium on VLSI, p.137, ACM, 2007.
[20] T. Schoellhammer, E. Osterweil, B. Greenstein, M. Wimbrow, and D. Estrin, "Lightweight temporal compression of microclimate datasets," Proc. 29th Annual IEEE International Conference on Local Computer Networks, pp.516–524, 2004.
[21] Y. Wang, Y. Hsieh, and Y. Tseng, "Compression and storage schemes in a sensor network with spatial and temporal coding techniques," Proc. Vehicular Technology Conference, VTC Spring, pp.148–152, 2008.
[22] T. Banerjee, K. Chowdhury, and D. Agrawal, "Tree based data aggregation in sensor networks using polynomial regression," Proc. 8th International Conference on Information Fusion, pp.1–6, 2005.
[23] H. Luo, Y. Tong, G. Pottie, and C. Los Angeles, "A two-stage DPCM scheme for wireless sensor networks," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, pp.1–6, 2005.
[24] A. Ciancio and A. Ortega, "A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, pp.1–6, 2005.
[25] http://www.pmel.noaa.gov/tao

Yongpan Liu was born in Henan Province, P.R. China. He received his B.S., M.S. and Ph.D. degrees from Electronic Engineering Department, Tsinghua University in 1999, 2002, and 2007. He worked as a research fellow in Tsinghua University from 2002 to 2004. Since 2007, he became an assistant professor in Electronic Engineering Department, Tsinghua University. He has published over 30 peer-reviewed conference and journal papers, supported by NSFC, 863, 973 Program. His main research interests include embedded systems, power-aware architecture and VLSI design and electronic design automation. Specifically, his projects consist of ultra-low power wireless sensor network and heterogeneous MPSoCs for software defined radio. He is an IEEE member and served as a reviewer and TPC of several IEEE conferences and TVLSI.

Shuangchen Li was born in Liaoning Province, P.R. China. He is now a junior undergraduate student and working towards his B.S. degree in the Department of Electronic Engineering, Tsinghua University. His main research interests are HW/SW co-design methodology and VLSI design.

Jue Wang was born in Anhui Province, P.R. China. She was a graduate student in the Department of Electronic Engineering, Tsinghua University. She is now with Marvell. Her main research interests are Low power VLSI designs and wireless sensor network.

Beihua Ying was born in Zhejiang Province, P.R. China. She was a Ph.D. student in the Department of Electronic Engineering, Tsinghua University. She is now with University. Her main research interests are Low power techniques for wireless sensor network.

Huazhong Yang was born in Sichuan Province, P.R. China, on Aug. 18, 1967. He received B.S., M.S., and Ph.D. Degrees in Electronic Engineering from Tsinghua University, Beijing, in 1989, 1993, and 1998, respectively. Now, he is a Professor and Head of the Institute of Circuits and Systems in Electronic Engineering Department, Tsinghua University, Beijing. His research interests include CMOS radio-frequency integrated circuits, VLSI system structure for digital communications and media processing, wireless sensor network, low-voltage and low-power circuits, and computer-aided design methodologies for system integration. He has authored and co-authored over 30 patents, 7 books, and over 200 journal and conference papers. He was granted National Palmary Young Researcher Fund of China in 2000 and the Special Governmental Subsidy from the Chinese State Council in 2007.