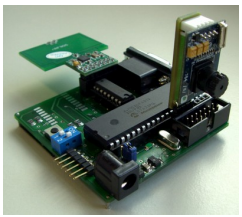
Wavelet Image Two-Line Coder (Wi2l) for Wireless Sensor Node with extremely little RAM

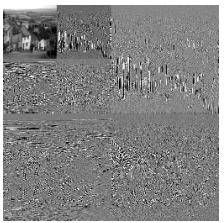
Stephan Rein, Stephan Lehmann, Clemens Gühmann
Wavelet Application Group
Technische Universität Berlin

to appear at dcc'09

Motivation

- Attributes for small wireless sensors: energy, scalibility, low-complexity, memory
- Idea: Design a wavelet coder for picture compression using not more than 2 kByte RAM for a 256x256x8 picture





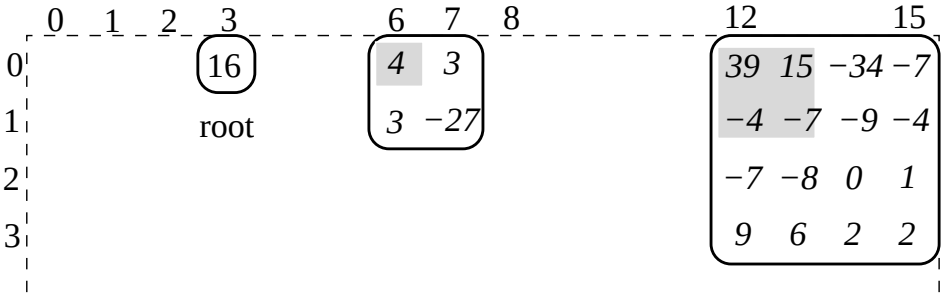
Contents

1. Related Work

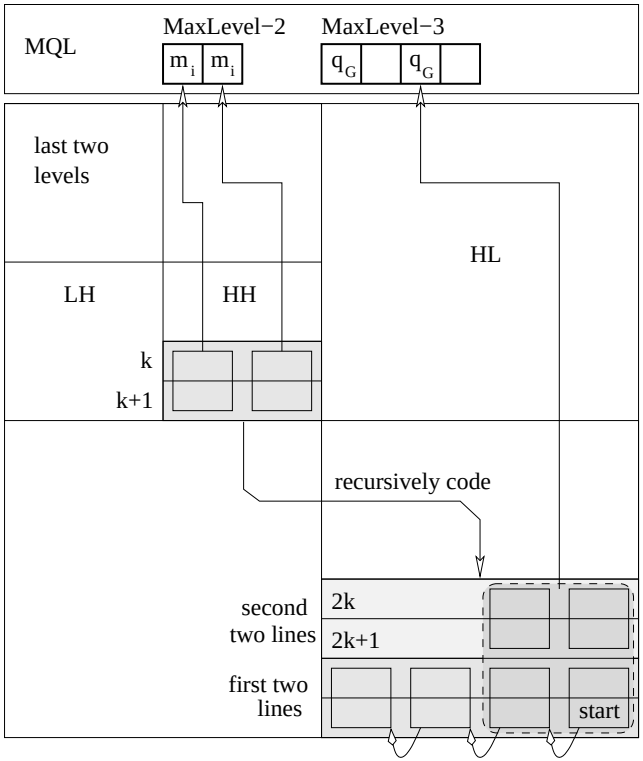
- Work exists for FPGAs or DSPs, but not for microcontrollers
- [Lehmann et al., Sensor node filesystem , *Mobimedia'08*]
  $\Rightarrow$ access blocks of 512 bytes
- [Rein et al., Fractional wavelet filter for wireless sensor, *Mobimedia'08*]
  $\Rightarrow$ does the transform with 1.5 kByte
- [Guo et al., A fast and low complexity image codec based on backward coding of wavelet tree, *dcc'06*]
  $\Rightarrow$ needs 20 kByte

2. Notation

- m_i is a maximum quantization level (MQL) of four coefficients and all tree descendants
- q_{Gi} is a maximum quantization level of 16 coefficients and all tree descendants

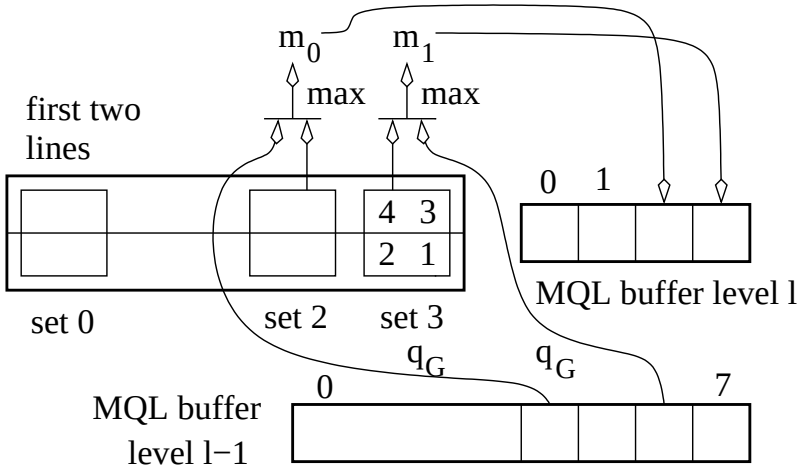


3. Wi2l Encoding Algorithm



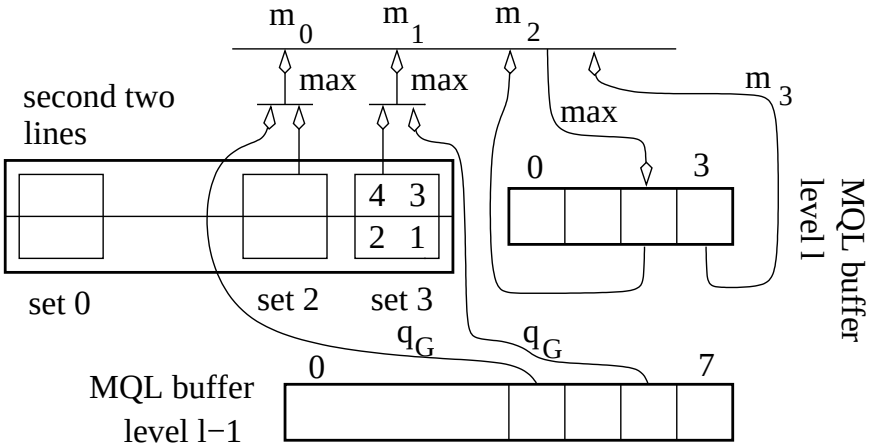
- *Code2Lines()*
- Recursion
- 126 bytes MQL buffer
- Subbands **HL, LH, HH**

3.1. Coding of first two lines



- Compute $m_i \Rightarrow$ encode the q_G of the previous level and 4 coefficients
- Retrieve MQL of previous tree coefficients through the MQL buffer, which was filled by recursion
- Store m_i in the MQL buffer of current level

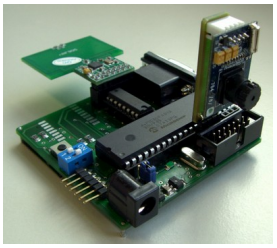
3.2. Coding of second two lines

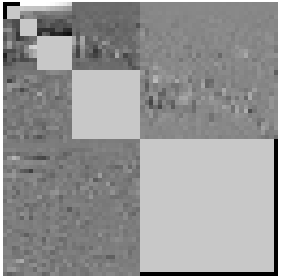


- Similarly: Compute $m_i \Rightarrow$ encode the q_G of the previous level and 4 coefficients
- However, retrieve m_i from current MQL buffer to compute the q_G levels
- Write the q_G levels to the current level MQL buffer

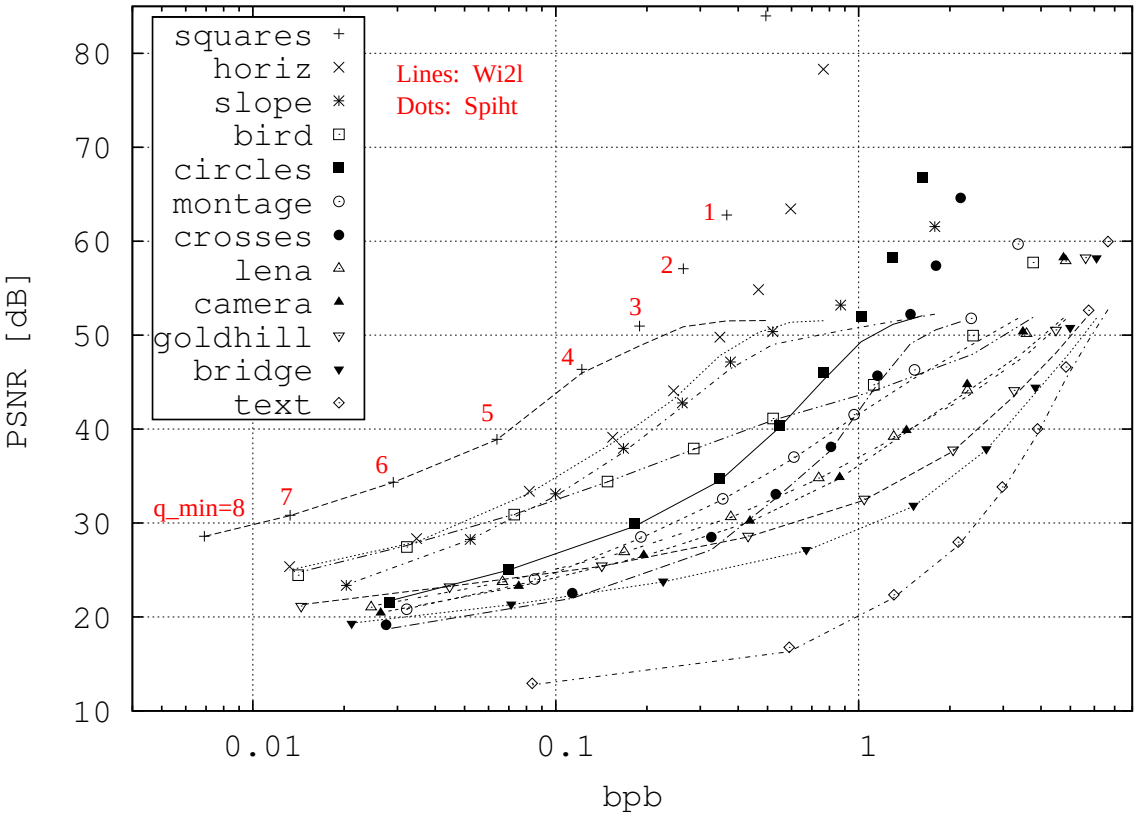
4. Results

- Compression performance:
  - Spiht coder from Said and Pearlman
  - Wi2l code in C
  - Fractional wavelet filter
- Time measurements:
  - own sensor with the Microchip dsPIC30F4013 with 2 kByte RAM and speed set to 29.491 MIPS
  - 64 MByte MMC-card connected to the controller

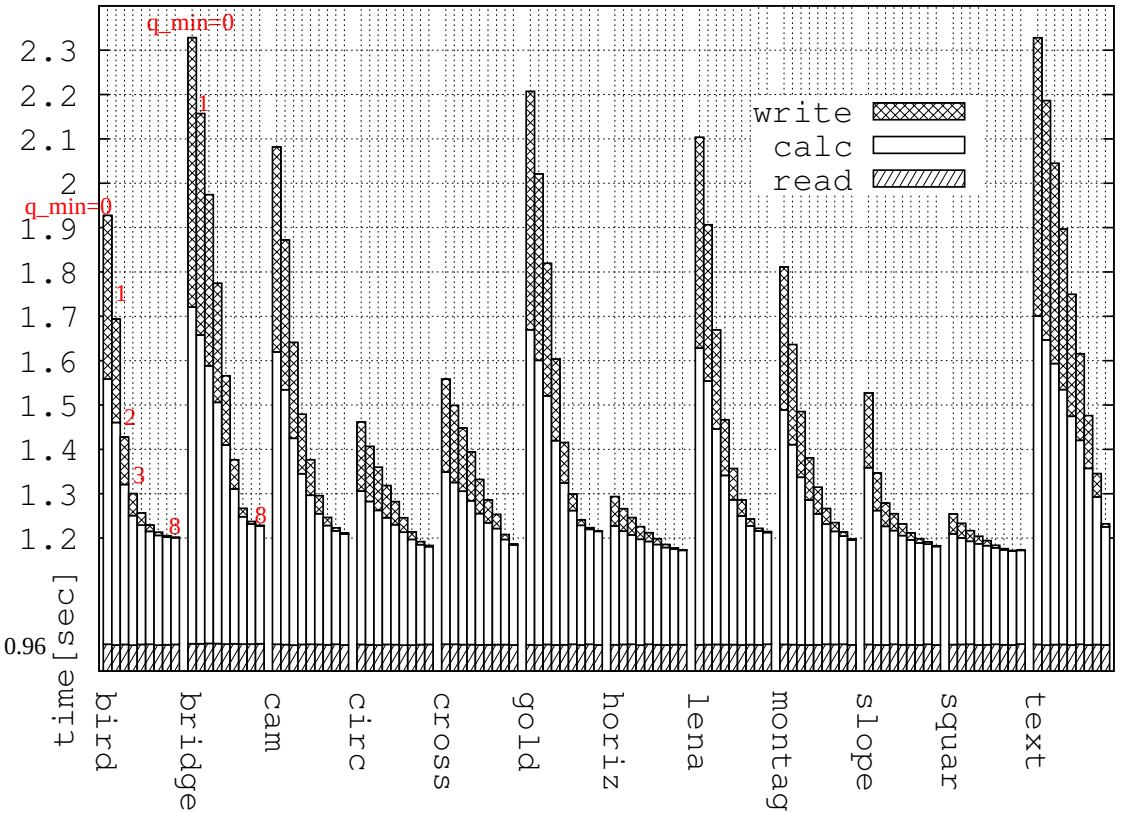



4.1. Results: Compression performance compared to Spiht



4.2. Results: Encoding times



5. Conclusion

- Wi2l needs less than 1.5 kBytes RAM: an input lines buffer of 512 bytes, a 512 byte binary buffer, and a 126 bytes MQL buffer
- Reads data line by line from a MMC-card in blocks of 512 bytes
- Exactly the same compression than Spiht
- Flexibility feature: Any typical sensor node (with UART and SPI) node can be extended
- Encoding times of 2 seconds, decoding in the range of 10 seconds
- Future work: Progressive feature

Thanks! Questions?

(Meanwhile see how Wi2l on our sensor is controlled.)