AEC/APC Symposium XXI

Improving Data Quality in Embedded Sensor Systems for APC

Julien Amelot, NIST
julien.amelot@nist.gov

Ya-Shian Li-Baboud, NIST
ya-shian.li-baboud@nist.gov
Goals

To characterize data quality for factory systems and develop fundamental information technology and engineering methods to improve next-generation factory APC applications.

Knowing is half the battle
NGF cost & cycle time reduction:

- Deterministic data availability and analysis
- Timely, accurate knowledge enables rapid real-time advanced process control and minimizing/eliminating false positives/negatives
- The faster the problems are detected and resolved, the less equipment use and time are wasted
Data Quality Goals

Time-Stamping
- Within nanoseconds of event occurrence

Data Sampling
- Deterministic sampling
- No data loss
- 10 000 Hz and beyond

Understand embedded resources impact
- Design needs to meet data sampling requirements
- Limitations given embedded resources available
- Optimize data sampling

To experimentally evaluate tradeoffs in system design features to meet data quality objectives for next-generation APC applications.
Overview of Sensor Network Testbed
Sensor Network

Requirements

- Time stamping within sub microsecond accuracy
- Deterministic 200 kHz data sampling

Results

- Robot sound localization within accuracy of 10 cm
- Ability to handle 4 microphones (sensors) at 200 kHz sampling rate

Relevance to APC data quality

- Requires time-stamping close to event occurrence
- High data sampling rates (APC: 100 Hz to 10 kHz)
- Requires minimal data loss for meaningful results (ideally no data loss)
- Minimize hardware cost
Embedded Hardware Design

- Analog to Digital Converter
  - 200KHz

- Field Programmable Gate Array
  - Spartan 3E
  - 100MHz

- CPU
  - Cirrus EP9302
  - ARM9
  - 200MHz
Embedded sensor data flow

- **STADIUM**
  - ADC: Up to 8 channels, sample data every 5us
  - Microphones

- **SPI BUS**
  - 1 data every 5us

- **FPGA**
  - IDI: Drives the bus
  - Timestamp

- **QWERK**
  - Kernel driver: Interruption runs every 500us, triggered by a CPU timer

- **LINUX**
  - User space driver: Collect data, build UDP packets, and send it

- **DECISIONAL NODE**
  - Run a UDP server to gather packets
  - Data filtering & data fusion
  - Multilateration
  - Pathfinding
  - Decision making
  - Robot driving

- **ETHERNET BUS**
Optimizing data sampling

Process

- Build a model of the system
- Characterize all critical parameters
- Identify the different constraints and bottleneck of the system
- Find the best set of parameters according to the needs

Data quality metrics considered

- Data sampling rate
- Data loss
- Data time-stamping latency/jitter
- Data processing latency/jitter
Model of the system (part of)

- **Number of sensors**
  - Sample rate per sensor (Hz) = \( \frac{\text{Data throughput}}{\text{SENSOR\_NUMBER}} \)

- **Define the SPI Bus Frequency**
  - SPI bus frequency (Hz) = \( \frac{10^8}{2 \times \text{SPIFREQ}} \)
  - System data throughput (data/sec) = \( \frac{\text{SPI bus frequency}}{18} \)

- **Define the size of the buffer in the FPGA**
  - Average size of UDP packets (data/packet) = \( \frac{\text{Data throughput}}{100} \)
  - FPGA’s buffer capacity (sec) = \( \frac{\text{RAMSIZE}}{\text{Data throughput}} \)

- **Define the Frequency for raising the interrupt to process data from FPGA buffer**
  - Kernel Interrupt raising period (sec) = \( \text{TIMERLOAD} \times 2.10^{-6} \)
  - Maximum number of data transferred per sec (FPGA → Linux) = \( \frac{\text{NB\_READ\_PER\_INT}}{\text{Kernel Interrupt raising period}} \)

Maximum number of data points processed during a single interrupt
Impact of the size of the buffer (RAM)

- RAM capacity of 8 data
- 128 data
- 64 data
- 32 data
- 16 data

The system cannot raise interrupts at such a frequency.

Data throughput too low.

FPGA's buffer becomes full.

Interrupts could be too long.

Number of values read per interrupt

Interrupt Period (s)
Impact of the speed of the Bus

System data throughput of 200 kHz

Number of values read per interrupt

Interrupt Period (s)

33 kHz
40 kHz
50 kHz
66 kHz
100 kHz

NIST
National Institute of Standards and Technology
Monitoring data loss

Value = f(time)

Monitoring of a 500 Hz sinusoid and monitoring of a 500 Hz sinusoid with no data loss with low data loss with heavy data loss

These 3 graphs represent the same data, in a different way
Optimizing data sampling parameters

- Good set of parameters
- Theoretical behavior
- High probability to overflow the FPGA’s buffer
- Worst case behavior actually measured
- Data throughput too low to transfer all the data
10KHz sampling case
CPU load effects

Raising Interrupt Period asked to the system (us)
Embedded data quality analysis

- **Kernel Interrupts**
  - Prioritizing data acquisition at the kernel level provides more reliable data processing rates

- **FPGA/FPGA Memory**
  - Alleviates requirements on CPU processing (e.g. interrupt frequency)
  - Timely processing of contextual information pertaining to the data (e.g. time stamps, uncertainty, etc.)

- **CPU Load**
  - Using non-preemptable interrupts, the CPU load is less of an issue; however, the non-preemptable interrupt may sacrifice performance of other processes running on the same CPU
  - Multiple non-preemptable processes running on the same CPU would adversely affect the performance of data sampling

- **Communication bus**
  - Data sampling rate is also limited by the size of the communication bus(es) between system components (e.g. FPGA and system processor).
Lessons Learned

- Ideally, design of embedded sensor device for APC requires assessment of requirements and limitations:
  - Measurement accuracy
  - Time-stamping accuracy
  - Data acquisition sampling rate
  - Data size
  - Data loss tolerance
  - Processing capability of systems in the data flow

- In designing equipment sensor devices, use of FPGA offers several benefits:
  - Allow accurate and reliable time-stamping
  - Time-stamping close to data measurement from the sensors
  - Provides hardware reliability, with software reconfigurability to meet changing needs

- Given limited resources, system tradeoffs can accommodate more stringent data acquisition requirements. For example,
  - If memory resources are low, interrupts will need to be more frequent
  - If system processor must manage a high level of priority interrupts, the design should allow for the memory buffer to store a larger quantity of data messages:
    - More memory
    - Improve data compression
Future Research

- **Additional tests comparing resource tradeoffs**
  - Kernel interrupts v. User space applications
  - FPGA buffer size
  - Multi-core architectures

- **Develop and test compression techniques**
  - To enable improved data quality under limited conditions

- **Develop improved data filtering / signal processing on the FPGA**
  - Potential to discard noisy data to reduce data processing volume for CPU

- **Extend simulation to address NGF-specific industry requirements**
  - Need to understand next-generation APC requirements in terms of data sampling metrics
  - Develop embedded design methods to address NGF needs
    - Multi-core design
    - Collaborate with University of Michigan on use of IEEE 1588 for time synchronization
References & Questions

- SEMI E151 Guide for Understanding Data Quality
- ISMI Next-Generation Factory Industry Briefing, O. Rothe, July 2009
- Data Quality and Time Synchronization, G. Crispieri, December 2006

Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States. Certain commercial equipment, instruments, or materials are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.