ASAM COMMON MDF 4.2.0

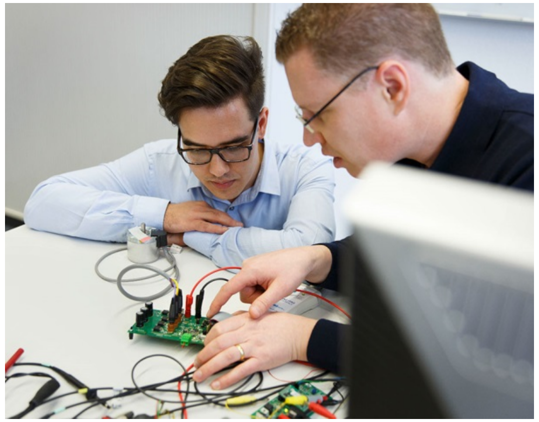
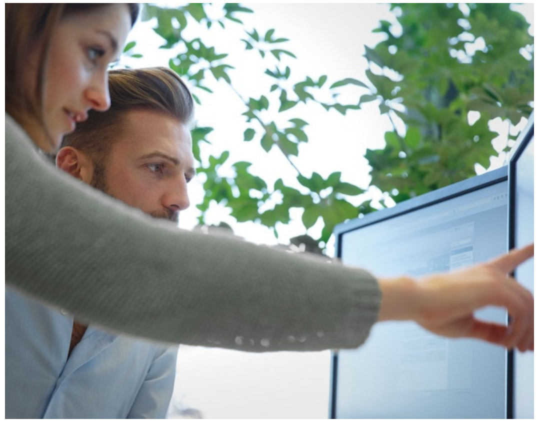
New Features and Enhanced Read Performance

Otmar Schneider, Vector Informatik GmbH

V0.1 | 2019-03-18

Vector Informatik GmbH

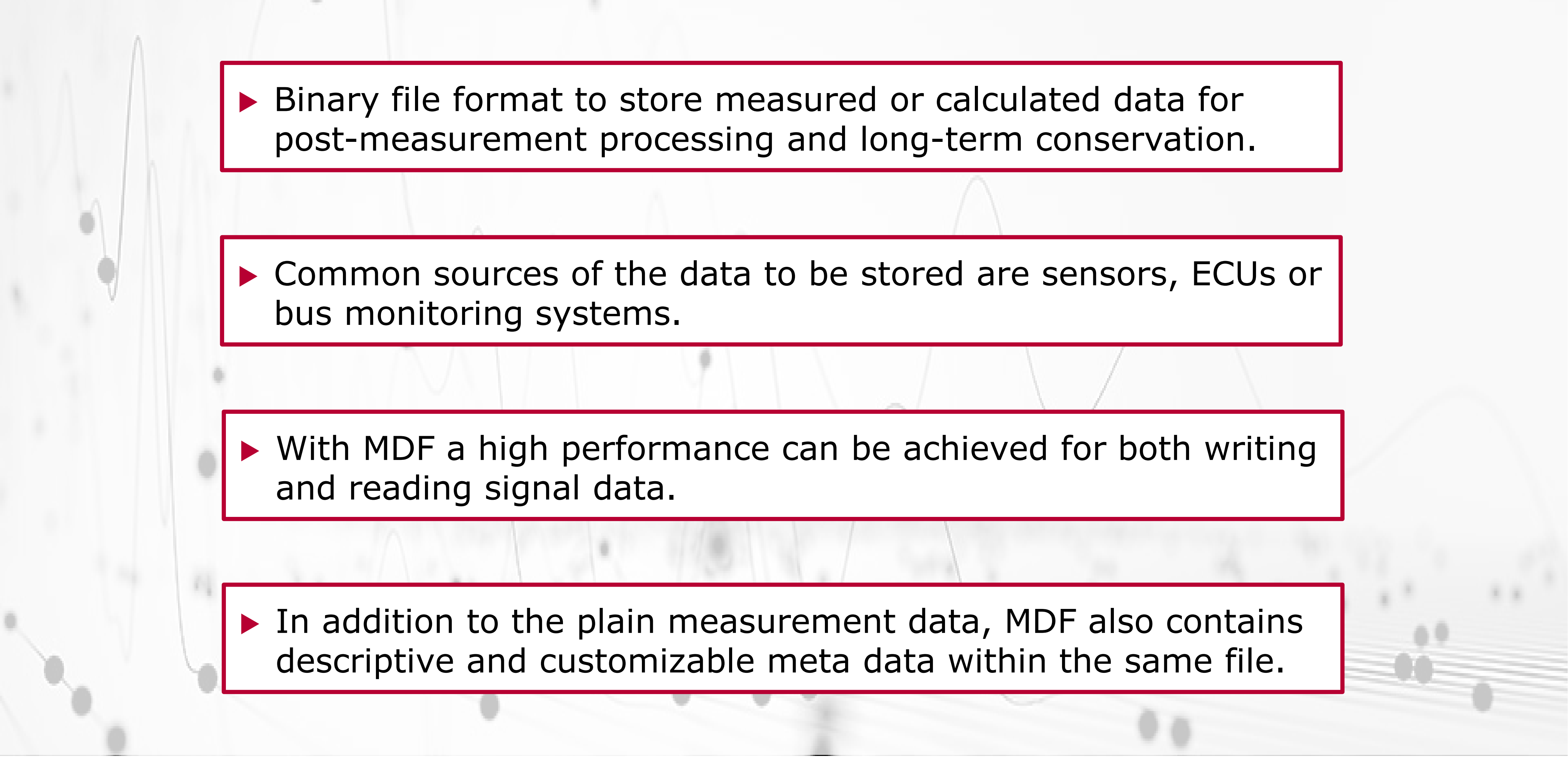
- Founded in 1988
- > 2,500 employees
- Headquarter in Stuttgart, Germany
- Subsidiaries: 26 locations in 12 countries
- Portfolio: hardware & software tools & engineering solutions for automotive industry (embedded, diagnostics, testing, measurement, calibration)
- Active in various standardization organizations, e.g. ASAM e.V.
- Supported ASAM standards: XCP, ASAP2, FIBEX, MDF, …

Agenda

1.	Welcome
2.	Introduction
3.	Motivation
4.	New Features in MDF 4.2
5.	Summary & Conclusion

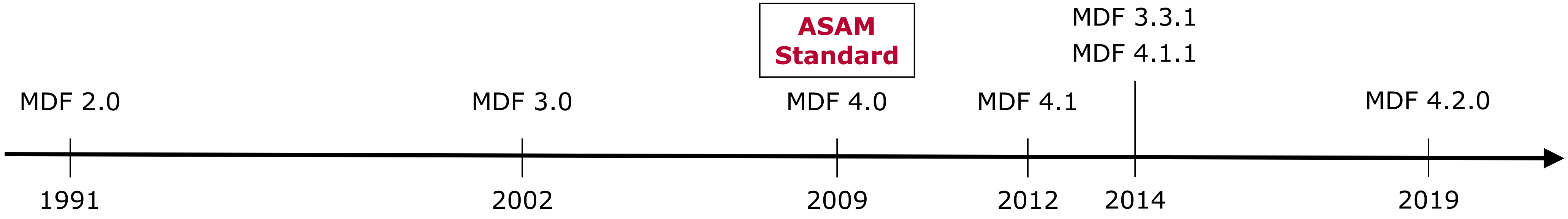
MDF (Measurement Data Format)



- Binary file format to store measured or calculated data for post-measurement processing and long-term conservation.
- Common sources of the data to be stored are sensors, ECUs or bus monitoring systems.
- With MDF a high performance can be achieved for both writing and reading signal data.
- In addition to the plain measurement data, MDF also contains descriptive and customizable meta data within the same file.

History

- 1990: MDF designed for use in the automotive industry
- 1991 until today: MDF versions 2.x and 3.x have successfully been used over many years and evolved to a de facto standard
- 2009: release of ASAM Common MDF 4.0.0 as result of a major update of the format and standardization by ASAM e.V.
- 2012: release of ASAM Common MDF 4.1.0 including three new associated standards
  - most important new features: compression of data, bus logging
- 2019: release of ASAM Common MDF 4.2.0
  - including new way to store data for enhanced read performance



Key Concepts of MDF

- Compact binary format organized in loosely coupled blocks
- Measurement data stored in records according to sampling rate
- Record layout and general signal description given by channels
- Supports multiple and non-periodic sample rates
- Synchronization via master channel concept
- Special data types and meta information used in automotive area
- Data received (e.g. from ECU) can be stored "as is"
- Conversion rules for calculation of physical values from stored raw values
- Extension of meta information by XML or "attachments"
  (embedding or linking of other files)

VECTOR

Agenda

1. Welcome

2. Introduction

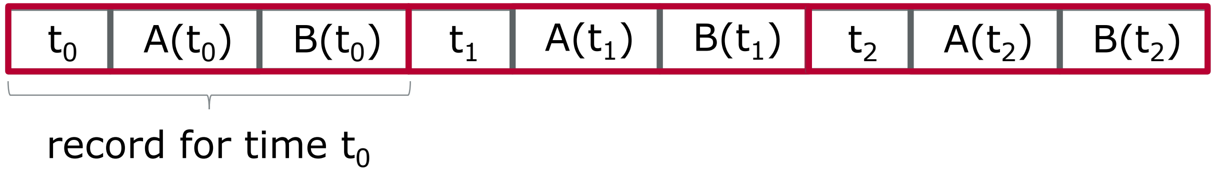
3. Motivation

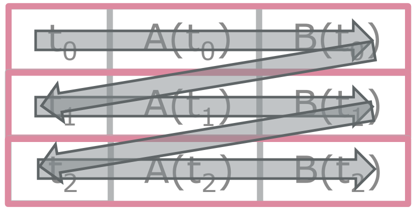
4. New Features in MDF 4.2

5. Summary & Conclusion

Why MDF 4.2?

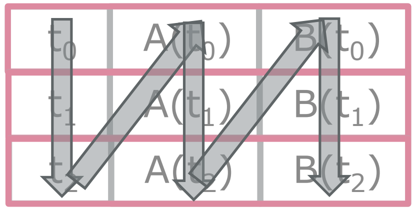
- MDF as storage format for measurement data management systems (e.g. ASAM ODS)
  - data often already delivered as MDF => avoid conversion
  - compact storage including meta data
- Feedback from developers
  - reading signal data is not optimal due to storage in "records"
  - each record typically contains a timestamp and values of the signals acquired simultaneously (same sampling rate / bus message)
  - record layout defined by channel group and contained channels
    - so-called "row-oriented" storage
    - ideal for writing
    - good for reading all signals values at a specific time t_n
    - but: not ideal for reading all values of a specific signal





Why MDF 4.2?

- Other formats use a "column-oriented" storage
  - all values of a signal are stored "en bloc"
  - ideal for fast reading (in most programming languages)



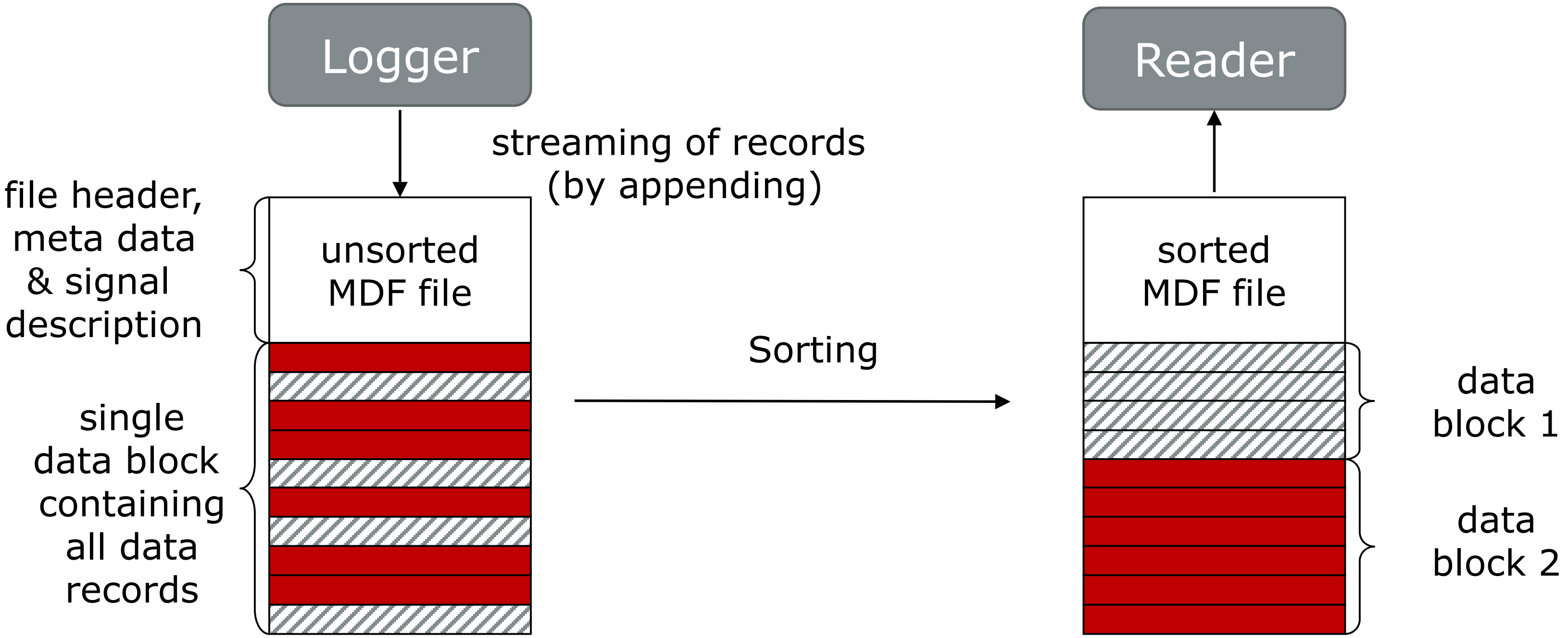
t_0	$A(t_0)$	$B(t_0)$
t_1	$A(t_1)$	$B(t_1)$
t_2	$A(t_2)$	$B(t_2)$

- Idea
  - introduce a new "flavor" of MDF which stores signal values in column-oriented storage
  - must contain identical information
  - simple (offline) transformation similar to "sorting" of MDF file for faster seek & reed

Write Once – Read Many

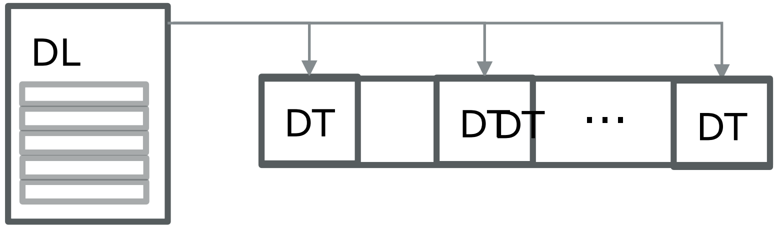
Previous Storage Variants

- MDF supports "unsorted" and "sorted" storage (since 1st version of MDF)
  - unsorted => easy to write by simply appending records (using a record ID)
  - sorted => easy to seek because only records of same type (and length) in a data block


Logger
streaming of records (by appending)
file header, meta data & signal description
unsorted MDF file
single data block containing all data records
Sorting
Reader
sorted MDF file
data block 1
data block 2

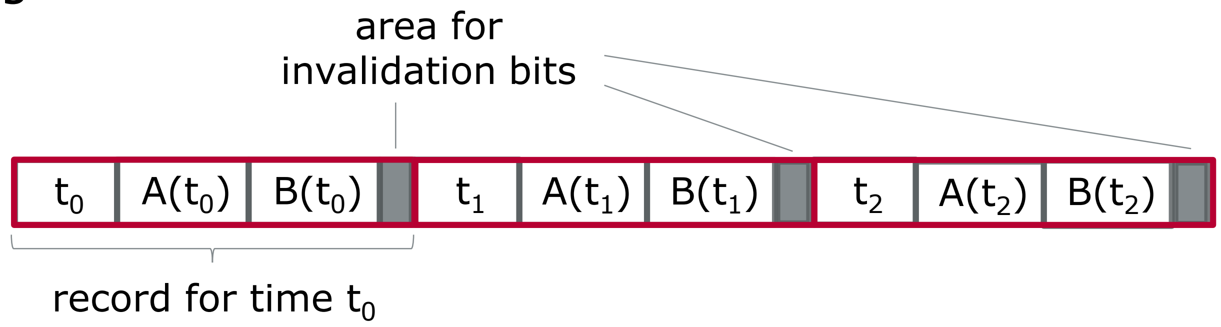
Other Storage Options

- MDF 4.0 introduced distributed data blocks
  - instead of a single data block per group there can be smaller ones referenced by a "list"
  - allows online writing of sorted MDF files without buffering all data or writing to temp files
- MDF 4.1 introduced compressed data blocks
  - allows compression of signal values
  - based on distributed data blocks



DT = data block
DL = data block list

- MDF 4.0 introduced "invalidation bits" in extra bytes at the end of the record
  - mark single value of a signal as "invalid"
  - reading all signal values and respective invalidation bit is clumsy
  - same problem as for row-oriented storage

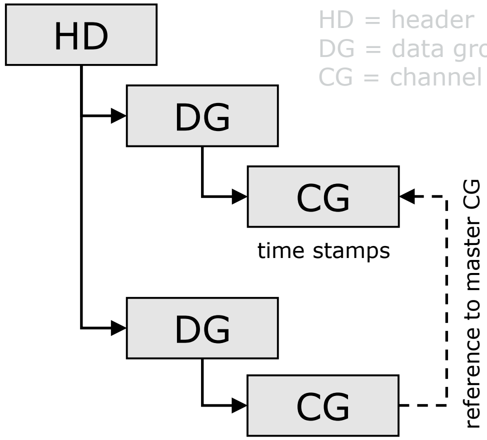


Agenda

1.	Welcome
2.	Introduction
3.	Motivation
4.	New Features in MDF 4.2
5.	Summary & Conclusion

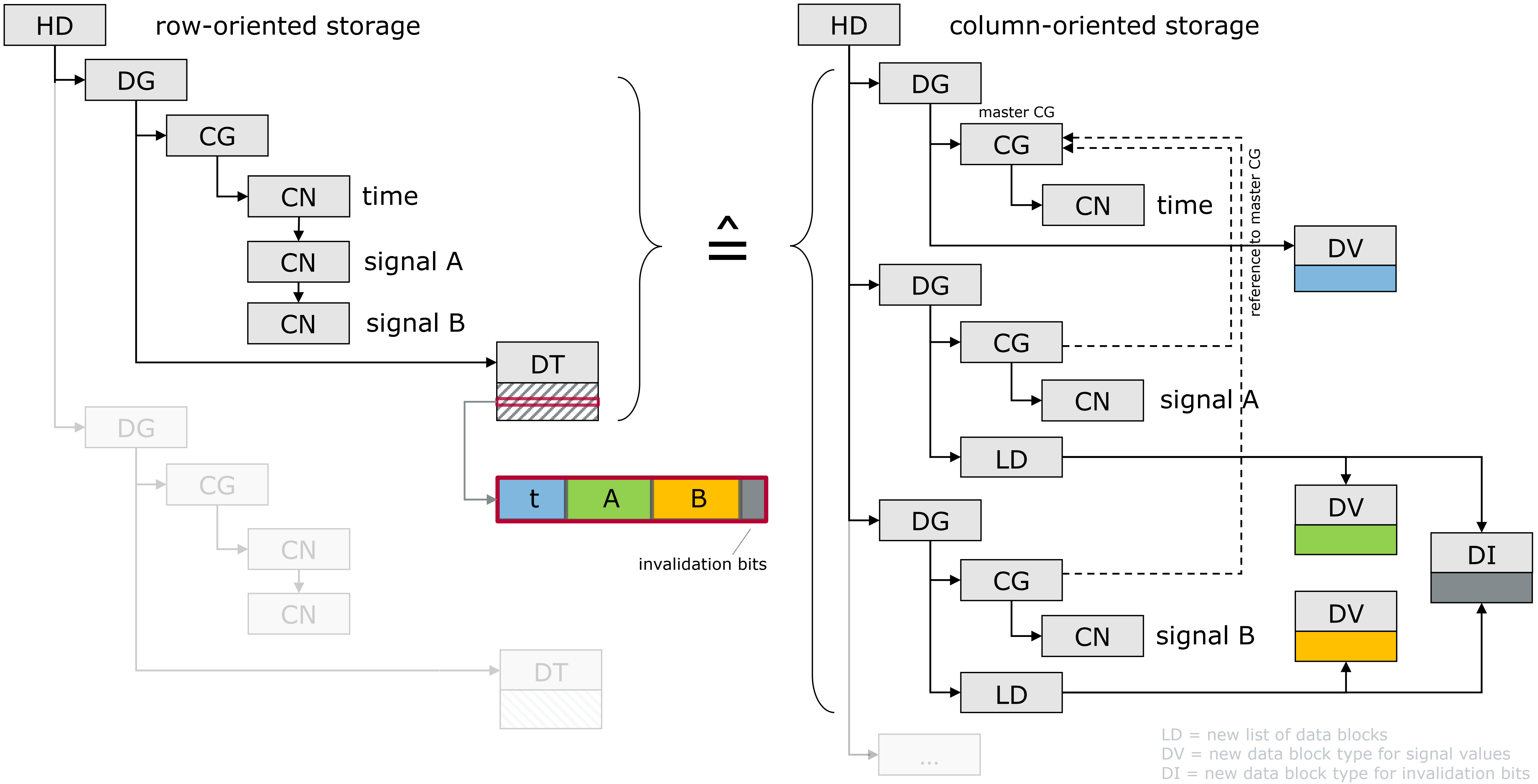
Column-Oriented Storage in MDF 4.2

- Introduced optional link to a "master" channel group
  - avoid duplication of master channel values (i.e. time stamps)
- Introduced new "list" and data block types
  - prevent that old tools read signal data without time stamps
  - however: meta data of the signals still readable!
- Store invalidation bits in a separate data block type
  - faster reading (like for signal values => all invalidation bytes are stored in a row)
  - this block can be omitted if there is no single "invalid" bit => no read at all!
- Column-oriented storage is achieved if there is only one channel per channel group
  - however: it is still allowed to store several channels per group if always read "together" (e.g. a structure or "complex number" with real and imaginary part)
- Concept still allows usage of compression and distributed data blocks
  - if only values of a one signal are stored, this often fits into a single data block



HD = header
DG = data group
CG = channel group

Block Structure for Column-Oriented Storage



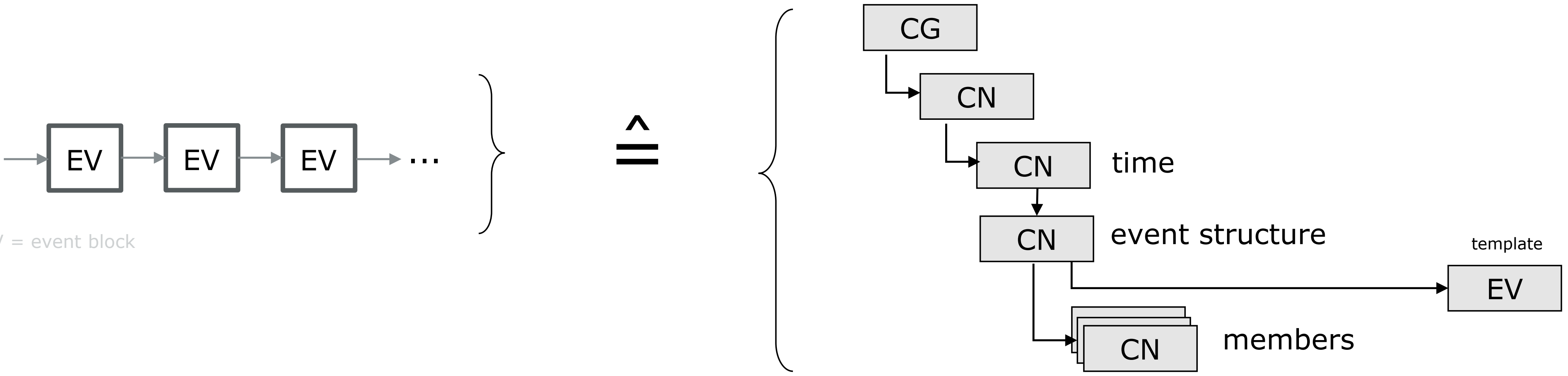
LD = new list of data blocks
DV = new data block type for signal values
DI = new data block type for invalidation bits

VECTOR

Further New Feature to Improve Read Performance

- Previously: events stored as linked list
  - OK for small number of events, but slow in case of large number (> 1000)
- MDF 4.2 offers an alternative way of storing events in channels ("event signals")
  - store events of same type in a structure and use a "template" event
  - channels are a proven mechanism to handle millions of samples
    - now open for events as well

=> loose a little bit of flexibility for the benefit of more efficient reading



EV = event block

Agenda

1. Welcome
2. Introduction
3. Motivation
4. New Features in MDF 4.2
5. Summary & Conclusion

Overview of New Features in MDF 4.2

- Column-Oriented Storage
  - New way to store signal data optimized for fast reading
  - No loss of information (signal values, meta data)
  - Compatibility: meta data still readable by old tools

- Event Signals
  - Efficiently store and read large number of events
  - Rely on same mechanism as reading ordinary signals

- Conclusion:
  - Improved Read Performance prepares MDF for "Big Data"
  - Fast acceptance and continued marked success expected

Questions?

For more information about Vector
and our products please visit

www.vector.com

Author:
Schneider, Otmar
Vector Germany