Error Detection And Correction (EDAC) Devices

Main Concepts used at the EDAC subsystem

There are several things to be aware of that aren't at all obvious, like *sockets, *socket sets*, *banks*, *rows*, *chip-select rows*, *channels*, etc...

These are some of the many terms that are thrown about that don't always mean what people think they mean (Inconceivable!). In the interest of creating a common ground for discussion, terms and their definitions will be established.

- Memory devices

The individual DRAM chips on a memory stick. These devices commonly output 4 and 8 bits each (x4, x8). Grouping several of these in parallel provides the number of bits that the memory controller expects: typically 72 bits, in order to provide 64 bits + 8 bits of ECC data.

- Memory Stick

A printed circuit board that aggregates multiple memory devices in parallel. In general, this is the Field Replaceable Unit (FRU) which gets replaced, in the case of excessive errors. Most often it is also called DIMM (Dual Inline Memory Module).

- Memory Socket

A physical connector on the motherboard that accepts a single memory stick. Also called as "slot" on several datasheets.

- Channel

A memory controller channel, responsible to communicate with a group of DIMMs. Each channel has its own independent control (command) and data bus, and can be used independently or grouped with other channels.

- Branch

It is typically the highest hierarchy on a Fully-Buffered DIMM memory controller. Typically, it contains two channels. Two channels at the same branch can be used in single mode or in lockstep mode. When lockstep is enabled, the cacheline is doubled, but it generally brings some performance penalty. Also, it is generally not possible to point to just one memory stick when an error occurs, as the error correction code is calculated using two DIMMs instead of one. Due to that, it is capable of correcting more errors than on single mode.

- Single-channel

The data accessed by the memory controller is contained into one dimm only. E. g. if the data is 64 bits-wide, the data flows to the CPU using one 64 bits parallel access. Typically used with SDR, DDR, DDR2 and DDR3 memories. FB-DIMM and RAMBUS use a different concept for channel, so this concept doesn't apply there.

- Double-channel

The data size accessed by the memory controller is interlaced into two dimms, accessed at the same time. E. g. if the DIMM is 64 bits-wide (72 bits with ECC), the data flows to the CPU using a 128 bits parallel access.

- Chip-select row

This is the name of the DRAM signal used to select the DRAM ranks to be accessed. Common chip-select rows for single channel are 64 bits, for dual channel 128 bits. It may not be visible by the memory controller, as some DIMM types have a memory buffer that can hide direct access to it from the Memory Controller.

- Single-Ranked stick

A Single-ranked stick has 1 chip-select row of memory. Motherboards commonly drive two chip-select pins to a memory stick. A single-ranked stick, will occupy only one of those rows. The other will be unused.

- Double-Ranked stick

A double-ranked stick has two chip-select rows which access different sets of memory devices. The two rows cannot be accessed concurrently.

- Double-sided stick

DEPRECATED TERM, see :ref:`Double-Ranked stick <doubleranked>`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]edac.rst`, **line 90**); ***backlink***
>
> Unknown interpreted text role "ref".

A double-sided stick has two chip-select rows which access different sets of memory devices. The two rows cannot be accessed concurrently. "Double-sided" is irrespective of the memory devices being mounted on both sides of the memory stick.

- Socket set

All of the memory sticks that are required for a single memory access or all of the memory sticks spanned by a chip-select row. A single socket set has two chip-select rows and if double-sided sticks are used these will occupy those chip-select rows.

- Bank

This term is avoided because it is unclear when needing to distinguish between chip-select rows and socket sets.

Memory Controllers

Most of the EDAC core is focused on doing Memory Controller error detection. The :c:func:`edac_mc_alloc`. It uses internally the struct `mem_ctl_info` to describe the memory controllers, with is an opaque struct for the EDAC drivers. Only the EDAC core is allowed to touch it.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]edac.rst, line 113); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]edac.rst, line 118)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/linux/edac.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]edac.rst, line 120)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/edac/edac_mc.h
```

PCI Controllers

The EDAC subsystem provides a mechanism to handle PCI controllers by calling the :c:func:`edac_pci_alloc_ctl_info`. It will use the struct :c:type:`edac_pci_ctl_info` to describe the PCI controllers.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]edac.rst, line 125); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]edac.rst, line 125); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]edac.rst, line 129)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/edac/edac_pci.h
```

EDAC Blocks

The EDAC subsystem also provides a generic mechanism to report errors on other parts of the hardware via :c:func:`edac_device_alloc_ctl_info` function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]edac.rst, line 134); *backlink*

Unknown interpreted text role "c:func".

The structures :c:type:`edac_dev_sysfs_block_attribute`, :c:type:`edac_device_block`, :c:type:`edac_device_instance` and :c:type:`edac_device_ctl_info` provide a generic or abstract 'edac_device' representation at sysfs.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]edac.rst, line 137); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]edac.rst, line 137); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]edac.rst, line 137); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]edac.rst, line 137); *backlink*

Unknown interpreted text role "c:type".

This set of structures and the code that implements the APIs for the same, provide for registering EDAC type devices which are NOT standard memory or PCI, like:

- CPU caches (L1 and L2)
- DMA engines
- Core CPU switches
- Fabric switch units
- PCIe interface controllers
- other EDAC/ECC type devices that can be monitored for errors, etc.

It allows for a 2 level set of hierarchy.

For example, a cache could be composed of L1, L2 and L3 levels of cache. Each CPU core would have its own L1 cache, while sharing L2 and maybe L3 caches. On such case, those can be represented via the following sysfs nodes:

```
/sys/devices/system/edac/..

pci/            <existing pci directory (if available)>
mc/             <existing memory device directory>
cpu/cpu0/..     <L1 and L2 block directory>
        /L1-cache/ce_count
                 /ue_count
        /L2-cache/ce_count
                 /ue_count
cpu/cpu1/..     <L1 and L2 block directory>
        /L1-cache/ce_count
                 /ue_count
        /L2-cache/ce_count
                 /ue_count
...

the L1 and L2 directories would be "edac_device_block's"
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]edac.rst, line 178)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/edac/edac_device.h
```