PCI Configuration Registers (CSRs)

Libmsr Version 0.2.1

Libmsr versions 2.0+ support PCI Configuration Registers. There are many performance counters for uncore Intel devices, such as the iMC (integrated memory controller), QPI (quick path interconnect), and PCU (power control unit). These registers are similar to MSRs but have a completely different way of being accessed. We only provide a batch interface to these registers as pread/pwrite don't make sense. This functionality also REQUIRES CSR-SAFE. Modifying CSRs is too dangerous to allow unchecked modification, you have been warned.

Initialize Libmsr CSR component

Initialize Libmsr CSR component

```
init_csr();
```

iMC registers

These registers allow you to count performance events with the integrated memory controller, which gives useful performance information relating to memory use.

1. Initialize Libmsr CSR component if not done already
2. Initialize the counters you need
3. Specify what the counters should do. Some functions have been created for frequently used tasks.
4. Print out the data with the respective dump function or access raw data with storage function.

Using iMC CSRs

```
init_csr();

init_imc_ctrs();

// iMC counter 0 will count all memory bandwidth.

// Valid registers are 0-3 and valid constrains are

// ALL_BW, READ_BW, and WRITE_BW

mem_bw_on_ctr(0, ALL_BW);

// Count read and write instructions on counters 1 and 2 respectively.

mem_pct_rw_on_ctr(1, 2);


// dump the memory bandwidth in bytes on that counter.

dump_mem_bw_from_ctr(0, stdout);

dump_mem_pct_rw_from_ctr(1, 2, READ_PCT, stdout);

dump_mem_pct_rw_from_ctr(1, 2, WRITE_PCT, stdout);


// Alternatively, access raw data

// read the batch corresponding to counter 0

read_imc_counter_batch(0);

struct imc_ctrs_data *pcd = imc_ctr_storage();

int i;

// IMC_NUMCTRS defined in master.h

for (i = 0; i < IMC_NUMCTRS * num_sockets(); i++)

{
```

```
printf("counter value is %d\n", pcd->ctr0[i]);
}
```

Custom iMC Events

While there are many functions for commonly used metrics, there are thousands of events these registers can count. To use these you will have to read the uncore performance guide for your architecture to find what parameters to use in the setup functions.

1. Initialize Libmsr CSR component if not done already
2. Initialize the counters you need
3. Specify what the counters should do based on the events in the uncore performance guide. Macros have been created for many events/umasks.

using the imc_ctr_config function

```
// Arguments of the imc_ctr_config_function
int imc_ctr_config(uint32_t threshold, uint32_t ovf_en, uint32_t edge_det,
uint32_t umask, uint8_t event, const unsigned counter);
```

4. Read the data from the registers used and access the data

Custom iMC Events

```
// Read memory bandwidth on counter 2 with no increment threshold,
// no overflow flag, and no edge detection.
imc_ctr_config(0x0, 0x0, 0x0, UMASK_CAS_RD, EVT_CAS_COUNT, 2);


// read the data from iMC counter 2
read_imc_counter_batch(2);


// access the data
struct imc_ctrs_data *pcd = imc_ctr_storage();
int i;
// IMC_NUMCTRS defined in master.h
for (i = 0; i < IMC_NUMCTRS * num_sockets(); i++)
{
printf("counter value is %d\n", pcd->ctr2[i]);
}
```

QPI Registers

coming soon...

Related articles

- PCI Configuration Registers (CSRs)
- The Batch Interface