How to measure the DDR throughput

Contents

Stable: 02.08.2021 - 06:59 / Revision: 08.07.2021 - 13:59

Stable: 02.08.2021 - 08:00 | Revision: 30.07.2021 - 13:00

A quality version of this page, approved on *2 August 2021*, was based off this revision.

Contents

1 Article purpose

This article explains how to measure the overall DDR throughput generated during a Linux® user space command execution.

2 Measurement flow

The DDR throughput measurement is done via **counters** provided by the DDRPERFM internal peripheral and controlled by **stm32_ddr_pmu**[1] performance monitoring unit driver that is registered to the Linux kernel **perf framework**.

From user land side, the perf tool is used to interact with the perf framework in order to record given events (such as stm32_ddr_pmu) during a command execution.

Once recorded, the **stm32_ddr_pmu.py** Python script is used to compute the corresponding DDR throughputs.

Launching this script without any parameter returns the usage information:

```
Board $> python3 /usr/bin/stm32_ddr_pmu.py
Usage:
  python stm32_ddr_pmu.py [-d <ddr_freq>] -f <perf_file>
    -d ddr_freq: DDR frequency in MHz (533 MHz by default)
    -f perf_file: text file containing the output of
        perf stat -e stm32_ddr_pmu/read_cnt/,stm32_ddr_pmu/time_cnt/,stm32_ddr_pmu
/write_cnt/ -a -o <perf_file> <command>
```

Information

The stm32_ddr_pmu driver is compiled as a kernel module (stm32_ddr_pmu.ko) that may not be automatically probed at startup. If 'lsmod' command does not show this module, then make sure that ddrperfm node is enabled in the device tree and run the command 'modprobe stm32_ddr_pmu' to get it properly loaded.

3 Example

Let us see how to measure the DDR throughput generated during the 'sleep 1' command execution.

First, record the counters into 'perf.txt' file, providing the command to launch as parameter:

```
Board $> perf stat -e stm32_ddr_pmu/read_cnt/,stm32_ddr_pmu/time_cnt/,stm32_ddr_pmu
/write_cnt/ -a -o perf.txt sleep 1
```

Then, run the Python script to convert the result in throughput values:

```
Board $> python3 /usr/bin/stm32_ddr_pmu.py -f perf.txt
R = 8 MB/s, W = 2 MB/s, R&W = 9 MB/s (DDR @ 533 MHz)
```

These low figures are due to the fact that the measurement was done with Weston stopped.

4 References

- drivers/perf/stm32_ddr_pmu.c

Stable: 02.08.2021 - 06:58 / Revision: 08.07.2021 - 13:27

Contents

1 Article purpose

This article explains how to measure the overall DDR throughput generated during a Linux® user space command execution.

2 Measurement flow

The DDR throughput measurement is done via **counters** provided by the DDRPERFM internal peripheral and controlled by **stm32_ddr_pmu**[1] performance monitoring unit driver that is registered to the Linux kernel **perf framework**.

From user land side, the perf tool is used to interact with the perf framework in order to record given events (such as stm32_ddr_pmu) during a command execution.

Once recorded, the **stm32_ddr_pmu.py** Python script is used to compute the corresponding DDR throughputs.

Launching this script without any parameter returns the usage information:

```
Board $> python3 /usr/bin/stm32_ddr_pmu.py
Usage:
  python stm32_ddr_pmu.py [-d <ddr_freq>] -f <perf_file>
    -d ddr_freq: DDR frequency in MHz (533 MHz by default)
    -f perf_file: text file containing the output of
        perf stat -e stm32_ddr_pmu/read_cnt/,stm32_ddr_pmu/time_cnt/,stm32_ddr_pmu
/write_cnt/ -a -o <perf_file> <command>
```

Information

The stm32_ddr_pmu driver is compiled as a kernel module (stm32_ddr_pmu.ko) that may not be automatically probed at startup. If 'lsmod' command does not show this module, then make sure that ddrperfm node is enabled in the device tree and run the command 'modprobe stm32_ddr_pmu' to get it properly loaded.

3 Example

Let us see how to measure the DDR throughput generated during the 'sleep 1' command execution.

First, record the counters into 'perf.txt' file, providing the command to launch as parameter:

```
Board $> perf stat -e stm32_ddr_pmu/read_cnt/,stm32_ddr_pmu/time_cnt/,stm32_ddr_pmu
/write_cnt/ -a -o perf.txt sleep 1
```

Then, run the Python script to convert the result in throughput values:

```
Board $> python3 /usr/bin/stm32_ddr_pmu.py -f perf.txt
R = 8 MB/s, W = 2 MB/s, R&W = 9 MB/s (DDR @ 533 MHz)
```

These low figures are due to the fact that the measurement was done with Weston stopped.

4 References

- drivers/perf/stm32_ddr_pmu.c

Stable: 16.03.2021 - 15:55 / Revision: 16.03.2021 - 14:22

Contents

1 Article purpose

This article explains how to measure the overall DDR throughput generated during a Linux® user space command execution.

2 Measurement flow

The DDR throughput measurement is done via **counters** provided by the DDRPERFM internal peripheral and controlled by **stm32_ddr_pmu**[1] performance monitoring unit driver that is registered to the Linux kernel **perf framework**.

From user land side, the perf tool is used to interact with the perf framework in order to record given events (such as stm32_ddr_pmu) during a command execution.

Once recorded, the **stm32_ddr_pmu.py** Python script is used to compute the corresponding DDR throughputs.

Launching this script without any parameter returns the usage information:

```
Board $> python3 /usr/bin/stm32_ddr_pmu.py
Usage:
  python stm32_ddr_pmu.py [-d <ddr_freq>] -f <perf_file>
    -d ddr_freq: DDR frequency in MHz (533 MHz by default)
    -f perf_file: text file containing the output of
        perf stat -e stm32_ddr_pmu/read_cnt/,stm32_ddr_pmu/time_cnt/,stm32_ddr_pmu
/write_cnt/ -a -o <perf_file> <command>
```

Information

The stm32_ddr_pmu driver is compiled as a kernel module (stm32_ddr_pmu.ko) that may not be automatically probed at startup. If 'lsmod' command does not show this module, then make sure that ddrperfm node is enabled in the device tree and run the command 'modprobe stm32_ddr_pmu' to get it properly loaded.

3 Example

Let us see how to measure the DDR throughput generated during the 'sleep 1' command execution.

First, record the counters into 'perf.txt' file, providing the command to launch as parameter:

```
Board $> perf stat -e stm32_ddr_pmu/read_cnt/,stm32_ddr_pmu/time_cnt/,stm32_ddr_pmu
/write_cnt/ -a -o perf.txt sleep 1
```

Then, run the Python script to convert the result in throughput values:

```
Board $> python3 /usr/bin/stm32_ddr_pmu.py -f perf.txt
R = 8 MB/s, W = 2 MB/s, R&W = 9 MB/s (DDR @ 533 MHz)
```

These low figures are due to the fact that the measurement was done with Weston stopped.

4 References

- drivers/perf/stm32_ddr_pmu.c

Stable: 09.10.2019 - 15:37 / Revision: 04.09.2019 - 08:00

Contents

1 Article purpose

This article explains how to measure the overall DDR throughput generated during a Linux® user space command execution.

2 Measurement flow

The DDR throughput measurement is done via **counters** provided by the DDRPERFM internal peripheral and controlled by **stm32_ddr_pmu**[1] performance monitoring unit driver that is registered to the Linux kernel **perf framework**.

From user land side, the perf tool is used to interact with the perf framework in order to record given events (such as stm32_ddr_pmu) during a command execution.

Once recorded, the **stm32_ddr_pmu.py** Python script is used to compute the corresponding DDR throughputs.

Launching this script without any parameter returns the usage information:

```
Board $> python3 /usr/bin/stm32_ddr_pmu.py
Usage:
  python stm32_ddr_pmu.py [-d <ddr_freq>] -f <perf_file>
    -d ddr_freq: DDR frequency in MHz (533 MHz by default)
    -f perf_file: text file containing the output of
        perf stat -e stm32_ddr_pmu/read_cnt/,stm32_ddr_pmu/time_cnt/,stm32_ddr_pmu
/write_cnt/ -a -o <perf_file> <command>
```

Information

The stm32_ddr_pmu driver is compiled as a kernel module (stm32_ddr_pmu.ko) that may not be automatically probed at startup. If 'lsmod' command does not show this module, then make sure that ddrperfm node is enabled in the device tree and run the command 'modprobe stm32_ddr_pmu' to get it properly loaded.

3 Example

Let us see how to measure the DDR throughput generated during the 'sleep 1' command execution.

First, record the counters into 'perf.txt' file, providing the command to launch as parameter:

```
Board $> perf stat -e stm32_ddr_pmu/read_cnt/,stm32_ddr_pmu/time_cnt/,stm32_ddr_pmu
/write_cnt/ -a -o perf.txt sleep 1
```

Then, run the Python script to convert the result in throughput values:

```
Board $> python3 /usr/bin/stm32_ddr_pmu.py -f perf.txt
R = 8 MB/s, W = 2 MB/s, R&W = 9 MB/s (DDR @ 533 MHz)
```

These low figures are due to the fact that the measurement was done with Weston stopped.

4 References

- drivers/perf/stm32_ddr_pmu.c