etnaviv

The wonderful world of performance counters
Christian Gmeiner

Long term etnaviv developer

whoami
agenda

GPU performance counter

Vivante GPUs

binary blob

etnaviv

Future
Can help to analyze the performance and execution characteristics of applications.
Vivante GPUs
Limitations

No access via command stream to the performance counters exception: occlusion queries

There might be hardware support in coming GPUs

typedef enum _gceProbeStatus
{
    gcvPROBE_Disabled = 0,
    gcvPROBE_Paused = 1,
    gcvPROBE_Enabled = 2,
}
gceProbeStatus;

typedef enum _gceProbeCmd
{
    gcvPROBECMD_BEGIN = 0,
    gcvPROBECMD_PAUSE = 1,
    gcvPROBECMD_RESUME = 2,
    gcvPROBECMD_END = 3,
}
gceProbeCmd;

typedef struct _gcsPROBESTATES
{
    gceProbeStatus              status;
    gctUINT32                   probeAddress;
}gcsPROBESTATES;

https://source.codeaurora.org/external/imx/linux-imx
branch: imx_5.4.3_2.0.0
How to access performance counter

simple register read

mux config write + register read

per pixel pipe counter values

take care of clock gating and debug register access
galcore interface

gcvHAL_GET_PROFILE_SETTING

gcvHAL_SET_PROFILE_SETTING

gcvHAL_READ_PROFILER_REGISTER_SETTING

gcvHAL_READ_ALL_PROFILE_REGSITERS_PART1

gcvHAL_READ_ALL_PROFILE_REGSITERS_PART2

gcvHAL_PROFILE_REGSITERS_2D

https://source.codeaurora.org/external/imx/linux-imx branch: imx_5.4.3_2.0.0
galcore as register documentation

```c
/* HW profile information. */
typedef struct _gcsPROFILER_COUNTERS_PART1
{
gctUINT32    gpuTotalRead64BytesPerFrame;
gctUINT32    gpuTotalWrite64BytesPerFrame;
/* SH */
gctUINT32    ps_branch_inst_counter;
}
gcsPROFILER_COUNTERS_PART1;
```

gcmkONERROR(gckOS_WriteRegisterEx(Hardware->os, Hardware->core, 0x00470, (((gctUINT32) (0)) & ~(((gctUINT32) (((gctUINT32) (1 ? 31:24) & ~(((gctUINT32) (((gctUINT32) (((1 ? 31:24) + 1))))) << (0 ? 31:24)))) | (((gctUINT32) ((gctUINT32) (13) & ((gctUINT32) (((1 ? 31:24) + 1))))))) << (0 ? 31:24))));
gcmkONERROR(gckOS_ReadRegisterEx(Hardware->os, Hardware->core, 0x0045C, &profiler_part1->ps_branch_inst_counter));
```
etnaviv
How to support performance counter

Go the galcore way and provide an ioctl to readout all counter in one go

- quite easy to implement - see galcore
- different ioctl's per pipe (2D and 3D)
- ABI compatibility if new counters are supported
- no relation to a specific draw call
How to support performance counter

Emulate the missing counter access via command stream

- takes more time to get API/ABI right
- one interface for all pipes
- ABI compatibility if new counters are supported
- relation to a specific draw call
- we can support one-shot-readouts
Emulation via command stream

Extend the in-kernel command buffer to

- trigger an event aka IRQ
- stop the FE
- process the IRQ on the driver and some work
- restart the FE

This called a syncpoint in the etnaviv kernel driver.
kernel interface

There is a need to extend the current kernel interface to

- get a list of possible counters
- configure counter readout via submit ioctl
- expose counter's data to userspace
query domains and signals

```c
static const struct etnaviv_pm_domain doms_3d[] = {
    {
        .name = "SH",
        .profile_read = VIVS_MC_PROFILE_SH_READ,
        .profile_config = VIVS_MC_PROFILE_CONFIG0,
        .nr_signals = 9,
        .signal = (const struct etnaviv_pm_signal[]) {
            {
                "SHADER_CYCLES",
                VIVS_MC_PROFILE_CONFIG0_SH_SHADER_CYCLES,
                &perf_reg_read
            },
            ...
        }
    },
    ...
};
```
query domains and signals

```c
struct drm_etnaviv_pm_domain {
    __u32 pipe;       /* in */
    __u8  iter;       /* in/out, select pm domain at index iter */
    __u8  domain;     /* in, pm domain index */
    __u8  pad;
    __u16 iter;       /* in/out, select pm source at index iter */
    __u16 id;         /* out, id of signal */
    __u16 nr_signals; /* out, how many signals does this domain provide */
    char   name[64];   /* out, name of domain */
};

struct drm_etnaviv_pm_signal {
    __u32 pipe;       /* in */
    __u8  domain;     /* in, pm domain index */
    __u8  id;         /* out, id of domain */
    __u16 nr_signals; /* out, how many signals does this domain provide */
    char   name[64];   /* out, name of domain */
};

#define DRM_ETNAVIV_PM_QUERY_DOM       0x0a
#define DRM_ETNAVIV_PM_QUERY_SIG       0x0b
```
extending submit ioctl

```c
/* performance monitor request (pmr) */
#define ETNA_PM_PROCESS_PRE             0x0001
#define ETNA_PM_PROCESS_POST            0x0002
struct drm_etnaviv_gem_submit_pmr {
  __u32 flags; /* in, when to process request (ETNA_PM_PROCESS_x) */
  __u8  domain; /* in, pm domain */
  __u8  pad;
  __u16 signal; /* in, pm signal */
  __u32 sequence; /* in, sequence number */
  __u32 read_offset; /* in, offset from read_bo */
  __u32 read_idx; /* in, index of read_bo buffer */
};
```

Make it possible to store a counter value at a defined offset of a provided bo.

Also provide a way to define when the sampling should happen.
extending submit ioctl

Make it possible to provide multiple performance monitor requests (pmr).
Integration in gallium

each pipe_query has own bo

->begin_query(..): add a pmr to the submit with PM_PROCESS_PRE

->end_query(..): add a pmr to the submit with PM_PROCESS_POST

->get_query_result(..): cpu_prep(..), check sequence number, cpu_fini(..)

->flush(..) we do a end-flush-begin dance
Integration in gallium

GALLIUM_HUD

AMD_performance_monitor
Future goals
GPU utilization

Mostly work in progress
Bandwidth values

- FE_READ_BANDWIDTH
- MMU_READ_BANDWIDTH
- BLT_READ_BANDWIDTH
- SH0_READ_BANDWIDTH
- SH1_READ_BANDWIDTH
- PE_WRITE_BANDWIDTH
- BLT_WRITE_BANDWIDTH
- SH0_WRITE_BANDWIDTH
- SH1_WRITE_BANDWIDTH
perfetto

System profiling, app tracing and trace analysis

work in progress patches
Thank you!