Supreet Pal Singh

EVoC Student - Nouveau
Contents

● EVoC Project

● EVoC Experience and Suggestions
EVoC Project

Implementing a Software Scripting Engine on Fermi architecture based NVIDIA GPUs to achieve safe memory reclocking.
How did I get to this?

1. Web Developer
2. KDE contributor
3. No clue about X development
The path

- Onsite internships
  - Mozilla
  - Google
  - Apple
- GSoC deadline crossed
Two options

1. Android App Development
2. Nouveau
The Project

- Buying a new GPU - First NVIDIA card
- Fermi and Kepler
- Fermi memory reclocking
The problem with Fermi
nv50 laptops -> reclock memory and engines.

- Save power
- Default clock speed : Medium
nva3:
- load based reclocking
- Default clock speed: 1/3 to 1/2
- Low performance on Nouveau
FERMI

- Default clock speed: 10%!!
- Miserable performance
Process of Reclocking

- nv50 style
- Put card off the bus
- wait and write MMIO registers
The main issue

- nv50 used HWSQ (HardWare SeQuencer)
- HWSQ removed on Fermi
- Replaced by PDAEMON
PDAEMON

- Full access to the registers
- Capable of IRQs
- Used for Hardware monitoring and Reclocking
- ISA: $F\mu C$ (flexible microcode)
Open-Source PDAEMON

- Work done by Martin Peres ~mupuf
  - Host -> PDAEMON Communication
  - Fan Management
  - Works on nva3 to nvd9
  - Should work on Kepler
My Proposed Work

1. PDAEMON -> Host Communication
2. HWSQ replacement
3. Documentation
PDAEMON -> Host

- Ring Buffer
  - *GET / *PUT
  - *PUT writes
  - *GET reads

- Each process sends 4 params
  1. Process Id
  2. Message Id
  3. Payload Size
  4. Payload pointer
Basic checks

- Stop writing if buffer not read
- Stop reading if buffer empty
- Do not read if writing not complete
- Write if reading not complete
- Wrap around
Status

- PDAEMON -> HOST
  - TESTED
  - MERGED
Fermi Scripting Engine (FSE)

- HWSQ replacement
- Capable of memory reclocking
FSE Implementation Process

1. Understanding HWSQ
2. Designing the ISA
3. Implementing it in FμC
FSE Design

1. Full range Delay
2. Short range Delay
3. MMIO write
4. MMIO mask
5. MMIO wait
6. PDAEMON -> HOST message
Delay Implementation

- Short range:
  - 16bit Nano seconds
  - 16bit Micro Seconds
- Full range
  - 64bit Nano seconds
- Write
  - 8bit and 32bit
- Mask
- Wait
Send_msg

- Hooks up with PDAEMON->Host
- Takes two params
  - SIZE
  - MESSAGE
Unexpected Hurdle

- Planned demo for XDC
- Unaligned memory access
- Implemented ld_32, ld_16 and ld_08
Current Status

- Most of it tested and working
- Send_msg needs to support "msg_id"
- Send_msg needs pass testing
Documentation

1. Blogpost introducing Nouveau basics
2. Complete EVoC documentation on blog
3. Intro.txt by mwk in envytools
4. More Documentation for Newbies!
   // Beginner's Guide to KDE Development
Wrap Up

1. PDAEMON -> HOST :success
2. FSE : send_msg testing left
3. Documentation - Intro.txt & blogpost
Endless Vacation of Code
Propose a 13 week (3 Month) Project
$5000
  - $1000 upfront
  - $2000 mid-term
  - $2000 completion

Can start anytime
EVoC suggestions

- Flexibility == Good
- Need more specific rules != Refer GSoC
- Selection completely on Mentor
- PreRequisites on Wiki
- Open Mentors listed on Wiki
Thoughts on proposition by Martin

- Patch requirement compulsory?
- Limit a student to 2 EVoCs? NO?!
- Limit a student to 1EvoC/year? Yes.
- Upfront payment low? Yes.
- 3 Month engagement before project? No!
1. Something for Mentors?
2. PUBLICIZE!
Links

1. https://gitorious.org/pdaemon
2. supreetpal.blogspot.com
3. IRC nick: supreet
4. Email : supreetpal@gmail.com