Refactoring backlight and spi helpers in drm/tinydrm

OUTREACHY INTERNSHIP REPORT

Meghana Madhyastha
Outline

• About me
• Introduction
  – Project Goals
  – DRM
  – TinyDRM
• Backlight
• SPI
• Conclusion
About Me

• Round 15 (Dec 2017-Feb 2018) Outreachy intern
• Mentored by Daniel Vetter, Sean Paul and Noralf Trønnes to contribute to the drm subsystem.
Project Goals

• Refactor Backlight and SPI helpers in drm/tinydrm
• Make the helpers less tinydrm specific and make them generic so that they can be used by other drivers
Introduction: DRM

- Direct Rendering Manager
- Subsystem of the Linux kernel
- Exposes an API that user space programs can use to send commands and data to the GPU.
- Addresses limitation of fbdev: able to handle modern 3D accelerated GPU based video hardware
Introduction: DRM
## Introduction: Tinydrm

1. Driver helpers for very simple display hardware.
2. DRM drivers that are so small they can fit in a single source file.
3. Helpers for MIPI Display Bus Interface (DBI) compatible display controllers.
4. MIPI DBI implementation types:
   a. Motorola 6800 type parallel bus
   b. Intel 8080 type parallel bus
   c. SPI type with 3 options:
Introduction: Tinydrm

Tinydrm

---

Tinydrm drivers

tinydrm-core.c
tinydrmHelpers.c
tinydrm-pipe.c
Introduction: Tinydrm

Task: Refactor and move helpers from tinydrm-helpers to general drm source code files so that they can be used by other drivers.
Backlight

• Previously: Helpers present in tinydrm to find, enable and disable backlight
• The task: Backlight is used by other drivers in drm. Can we make the helpers general? Can we move them to video/backlight?
• During this process, I found that there was quite a bit of replicated code and different ways to enable and disable a backlight (different combinations of flags)
• Cleaned this up, made it more modular by encapsulating it into a backlight_enable and backlight_disable functions
Backlight

THEN

- tinydrm/helpers
  - Usage:
    
    ```c
    if (ddata->backlight) {
      ddata->backlight->props.power = FB_BLANK_UNBLANK;
      backlight_update_status(ddata->backlight);
    }
    
    (ENCAPSULATE THIS IN backlight_enable)
    ```

NOW

- video/backlight/backlight.c
  - Separate function for enabling and disabling backlight
    
    ```c
    - static inline int backlight_enable(struct backlight_device *bd)
    - static inline int backlight_disable(struct backlight_device *bd)
    - Usage: backlight_enable(ddata->backlight);
    ```
Backlight

THEN

- tinydrm_helpers
  - struct backlight_device
    *tinydrm_of_find_backlight(struct device *dev)
  - **Usage**: miipi->backlight = tinydrm_of_find_backlight(dev);

NOW

- video/backlight/backlight.c
  - struct backlight_device
    *of_find_backlight(struct device *dev)
  - **Usage**: miipi->backlight = of_find_backlight(dev);
SPI

- SPI: Interface bus - send data between microcontrollers and small peripherals (e.g., shift registers, sensors, and SD cards).
- In Tinydrm: Helpers for device drivers to communicate with spi.
• Goal: as part of my overall goal of refactoring, remove redundant chunk splitting in tinydrm spi helpers.
• Consider DMA transfers directly between the SPI hardware and a memory buffer
• The problem, we want to be able to send large >64kB buffers in one go to SPI.
• Tinydrm splits the buffer into max_dma_len chunks to spi-bcm2835 because drivers/spi/spi-bcm2835.c - has an upper bound check on dma transfer length (64KB) in bcm2835_spi_can_dma()
• Goal: 1) we want to remove splitting of buffer into small chunks in the tinydrm spi_helpers. This is because we want to leave it to the spi core to handle.
The solution

- Remove chunk splitting in tinydrm_spi_transfer in tinydrm-helpers and split the buffer in the driver (bcm2835).
- The spi core will split a buffer into max_dma_len chunks for the spi controller driver to handle.
- Remove the upper bound check on dma transfer length in bcm2835_spi_can_dma().
Remove the DMA length checking in spi-bcm2835.c

diff --git a/drivers/spi/spi-bcm2835.c b/drivers/spi/spi-bcm2835.c
index f35cc10772f6..0dccc45f158b8 100644
--- a/drivers/spi/spi-bcm2835.c
+++ b/drivers/spi/spi-bcm2835.c
@@ -365,19 +365,6 @@ static bool bcm2835_spi_can_dma(struct spi_master *master,
   if (tfr->len < BCM2835_SPI_DMA_MIN_LENGTH)
     return false;

- */ BCM2835_SPI_DLEN has defined a max transfer size as
- * 16 bit, so max is 65535
- * we can revisit this by using an alternative transfer
- * method - ideally this would get done without any more
- * interaction...
- */
- if (tfr->len > 65535) {
-   dev_warn_once(&spi->dev,
-       "transfer size of %d too big for dma-transfer\n",
-       tfr->len);
-   return false;
- }
-
“bcm2835_spi_transfer_one_message” in spi-bcm2835.c calls the helper spi_split_transfers_maxsize before calling spi_transfer_one_message to split the message into smaller chunks to be able to use dma.

Split the message into <64KB chunks
Conclusion

• Current state: The backlight patches have been accepted but the spi patches were still being discussed
• Refactored backlight and spi helpers
• Learned a lot about the linux kernel.
• Learned how to collaborate with people and communicate effectively.
QUESTIONS ?