Báo cáo Nghiên cứu Kỹ thuật Chuyên sâu: Kiến trúc và Triển khai Trình điều khiển Linux Kernel Thủ công (Manual DMA) cho Giao diện Unicam BCM2711 và Cảm biến IMX219 trên Raspberry Pi 4B

1. Tổng quan Điều hành và Phạm vi Nghiên cứu

Trong bối cảnh phát triển các hệ thống thị giác máy tính nhúng (embedded computer vision) hiệu năng cao, sự phụ thuộc vào các lớp trừu tượng hóa phần mềm (software abstraction layers) tiêu chuẩn thường trở thành điểm nghẽn về độ trễ và khả năng kiểm soát tài nguyên. Mặc dù hệ sinh thái Linux cung cấp các phân hệ mạnh mẽ như Video4Linux2 (V4L2) và khung quản lý bộ đệm videobuf2 (VB2) để đảm bảo tính tương thích phần cứng rộng rãi, các cơ chế này đưa vào hệ thống những chi phí xử lý không xác định (nondeterministic overheads), sự phân mảnh bộ nhớ và các xung đột tài nguyên tiềm ẩn khi hoạt động ở tần số quét cao.
Báo cáo này trình bày một phân tích toàn diện và thiết kế kiến trúc cho việc xây dựng một trình điều khiển nhân Linux (Linux kernel driver) tùy chỉnh dành cho System-on-Chip (SoC) Broadcom BCM2711, nền tảng của Raspberry Pi 4B. Mục tiêu cốt lõi là thiết lập cơ chế Truy cập Bộ nhớ Trực tiếp (Direct Memory Access - DMA) thủ công cho khối ngoại vi thu nhận hình ảnh CSI-2 (được gọi là Unicam), kết hợp với cảm biến hình ảnh Sony IMX219.
Bằng cách loại bỏ hoàn toàn sự phụ thuộc vào hàng đợi bộ đệm của videobuf2, giải pháp được đề xuất cho phép phân bổ các vùng bộ nhớ vật lý liền mạch tĩnh (sử dụng Contiguous Memory Allocator - CMA) và lập trình trực tiếp các thanh ghi DMA của khối Unicam. Phương pháp này tạo ra một đường ống dẫn dữ liệu (data pipeline) với độ trễ tối thiểu, loại bỏ chi phí quản lý trạng thái bộ đệm phức tạp và ngăn chặn các xung đột tài nguyên thường gặp trong các ngăn xếp video đa người dùng. Phân tích này đi sâu vào bản đồ thanh ghi (register map) của BCM2711 Unicam, kiến trúc định tuyến ngắt thông qua Bộ điều khiển Ngắt Chung (GIC-400), và các chuỗi lệnh I2C phức tạp cần thiết để vận hành cảm biến IMX219 ở chế độ khung hình cao 640x480.

2. Kiến trúc Phần cứng BCM2711 và Ngoại vi Unicam

2.1 Sự Tiến hóa của SoC BCM2711 và Không gian Địa chỉ

Raspberry Pi 4B đánh dấu bước chuyển mình quan trọng từ các dòng SoC BCM283x trước đây sang BCM2711. Mặc dù vẫn duy trì lõi đồ họa VideoCore VI, kiến trúc kết nối ngoại vi và bản đồ địa chỉ đã có những thay đổi căn bản ảnh hưởng trực tiếp đến việc phát triển trình điều khiển

mức thấp (low-level driver).
Một trong những thay đổi quan trọng nhất là việc giới thiệu chế độ "Low Peripherals". Trên các dòng Pi cũ (dựa trên BCM2835/6/7), địa chỉ bus của ngoại vi thường được ánh xạ từ 0x7E000000 sang địa chỉ vật lý ARM. Tuy nhiên, trên BCM2711, các thanh ghi ngoại vi được ánh xạ tới địa chỉ cơ sở vật lý của ARM là 0xFE000000.[1] Sự thay đổi này đòi hỏi sự chính xác tuyệt đối trong việc định nghĩa các macro truy cập thanh ghi trong mã nguồn kernel, vì việc sử dụng các offset cũ sẽ dẫn đến lỗi truy cập bộ nhớ (segmentation fault hoặc bus error).
Đối với phân hệ camera, BCM2711 tích hợp hai khối Unicam độc lập:

- **Unicam 0 (CSI0):** Thường được kết nối với cổng 2-lane trên các bo mạch Compute Module.
- **Unicam 1 (CSI1):** Giao diện 2-lane chính được lộ ra trên đầu nối của Raspberry Pi 4B tiêu chuẩn.[3]

Nghiên cứu xác định rằng trình điều khiển cần tập trung vào **Unicam 1**. Địa chỉ cơ sở kế thừa (legacy base address) của khối này là 0x7e801000, tương ứng với địa chỉ vật lý ARM là 0xfe801000 trên BCM2711.[4] Việc xác định chính xác địa chỉ này là bước tiên quyết để ánh xạ bộ nhớ I/O (ioremap) thành công.

2.2 Đặc tả Kỹ thuật của Ngoại vi Unicam CSI-2

Unicam không đơn thuần là một bộ điều khiển DMA thụ động. Đây là một khối phần cứng chức năng cố định (fixed-function hardware block) được thiết kế chuyên biệt để nhận dữ liệu nối tiếp tốc độ cao từ các cảm biến tuân thủ chuẩn MIPI CSI-2. Khả năng của Unicam bao gồm việc giải mã các gói dữ liệu CSI-2, tách dữ liệu hình ảnh khỏi các gói dữ liệu nhúng (embedded metadata), và thực hiện giải nén dữ liệu (ví dụ: chuyển đổi dữ liệu Bayer 10-bit thô thành các từ nhớ 16-bit) trước khi ghi vào bộ nhớ hệ thống.
Điểm khác biệt quan trọng cần lưu ý là Unicam hoạt động độc lập với bộ điều khiển DMA hệ thống (System DMA Controller) của BCM2711. Nó sở hữu động cơ DMA riêng biệt được điều khiển qua các thanh ghi chuyên dụng, cụ thể là:

- **IBSA0 (Image Buffer Start Address 0):** Địa chỉ bắt đầu của bộ đệm hình ảnh.
- **IBEA0 (Image Buffer End Address 0):** Địa chỉ kết thúc của bộ đệm hình ảnh.
- **IBWP (Image Buffer Write Pointer):** Con trỏ ghi hiện tại, cho biết vị trí dữ liệu đang được ghi vào SDRAM.

Cơ chế này cho phép Unicam hoạt động như một bộ đệm vòng (circular buffer) phần cứng. Khi con trỏ ghi đạt đến địa chỉ kết thúc (IBEA0), nó sẽ tự động quay lại địa chỉ bắt đầu (IBSA0) mà không cần sự can thiệp của CPU, miễn là các cờ ngắt và trạng thái được xử lý đúng cách. Đây là nền tảng cho việc triển khai "manual DMA" hiệu quả.

2.3 Cảm biến Sony IMX219 và Chế độ Analog Binning

IMX219 là cảm biến CMOS chiếu sáng sau (back-illuminated) với độ phân giải gốc 8-megapixel. Trong kiến trúc hệ thống, nó đóng vai trò là thiết bị I2C slave (địa chỉ 0x10) để nhận lệnh điều khiển và thiết bị MIPI CSI-2 master để truyền dữ liệu hình ảnh.
Đối với yêu cầu cụ thể về độ phân giải 640x480 (Mode 7) để đạt tốc độ khung hình cao (lên đến 200 fps hoặc hơn), cảm biến sử dụng chế độ "2x2 Analog Binning" đặc biệt.[6] Khác với

binning kỹ thuật số (digital binning) diễn ra sau khi chuyển đổi ADC, binning analog gộp điện tích từ các điểm ảnh lân cận ngay trên mảng cảm biến. Điều này mang lại hai lợi ích lớn:

1. **Tăng tốc độ đọc (Readout Speed):** Giảm số lượng hàng cần đọc, cho phép tăng FPS đáng kể.
2. **Cải thiện tỷ lệ Tín hiệu trên Nhiễu (SNR):** Tăng cường độ nhạy sáng mà không làm tăng nhiễu đọc.

Tuy nhiên, việc kích hoạt chế độ này đòi hỏi một chuỗi ghi thanh ghi I2C cực kỳ chính xác, khác biệt hoàn toàn so với các chế độ độ phân giải đầy đủ hoặc binning kỹ thuật số thông thường. Bất kỳ sai lệch nào trong các thanh ghi quy định thời gian khung hình (FRM_LENGTH), vùng cắt (cropping), hoặc chế độ binning (BINNING_MODE_H/V) đều có thể dẫn đến việc cảm biến không xuất dữ liệu hoặc dữ liệu không đồng bộ với bộ thu Unicam.

3. Phân tích Chiến lược: Bỏ qua Videobuf2 (VB2)

3.1 Hạn chế của Kiến trúc Videobuf2 Tiêu chuẩn

Khung làm việc videobuf2 là xương sống của hầu hết các trình điều khiển V4L2 hiện đại. Nó quản lý vòng đời của các bộ đệm video, từ việc cấp phát (reqbufs), xếp hàng (qbuf), đến việc lấy ra (dqbuf). Mặc dù mang lại sự an toàn và tính chuẩn hóa, VB2 áp đặt những ràng buộc không mong muốn cho các ứng dụng thời gian thực cực đoan:

- **Chi phí Quản lý Hàng đợi (Queue Management Overhead):** Mỗi thao tác xếp hàng và lấy hàng đều yêu cầu các lời gọi hệ thống (ioctl), xác nhận khóa (locking), và chuyển đổi ngữ cảnh giữa kernel-space và user-space.
- **Cấp phát Động (Dynamic Allocation Jitter):** VB2 thường dựa vào các cơ chế cấp phát động. Trong các hệ thống chạy lâu dài, sự phân mảnh bộ nhớ có thể dẫn đến thất bại trong việc cấp phát các bộ đệm lớn liền mạch, hoặc gây ra độ trễ không thể dự đoán.
- **Xung đột Tài nguyên:** Khi trình điều khiển tiêu chuẩn bcm2835-unicam được tải, nó chiếm quyền kiểm soát độc quyền đối với ngoại vi Unicam và đường ngắt. Để thực hiện DMA thủ công, bắt buộc phải vô hiệu hóa trình điều khiển tiêu chuẩn này thông qua Device Tree Overlay để tránh xung đột tài nguyên phần cứng (resource conflict).

3.2 Cơ chế Manual DMA: Giải pháp Zero-Copy

Giải pháp đề xuất thay thế VB2 bằng một mô hình quản lý bộ nhớ tĩnh và trực tiếp.

1. **Cấp phát Bộ nhớ Liền mạch (CMA):** Sử dụng dma_alloc_coherent để cấp phát một vùng bộ nhớ vật lý liền mạch duy nhất, kích thước lớn (ví dụ: đủ chứa 100 khung hình hoặc hoạt động như một bộ đệm vòng vô tận).
2. **Lập trình Thanh ghi Trực tiếp:** Ghi địa chỉ Bus (Bus Address) của vùng nhớ này trực tiếp vào các thanh ghi IBSA0 và IBEA0 của Unicam.
3. **Vòng lặp Vô tận Phần cứng:** Thiết lập Unicam để ghi liên tục vào vùng nhớ này. Trình điều khiển chỉ cần giám sát con trỏ IBWP hoặc xử lý ngắt Frame End để thông báo cho ứng dụng user-space biết vị trí dữ liệu mới nhất.
4. **Ánh xạ User-space:** Vùng nhớ CMA này được ánh xạ trực tiếp vào không gian địa chỉ của ứng dụng user-space thông qua mmap, cho phép truy cập dữ liệu hình ảnh mà

không cần bất kỳ thao tác sao chép nào (True Zero-Copy).

4. Quản lý Bộ nhớ và Dịch thuật Địa chỉ trên BCM2711

Một trong những khía cạnh phức tạp nhất khi làm việc với BCM2711 là sự tồn tại của nhiều không gian địa chỉ. Sự hiểu biết sai lệch về các không gian này là nguyên nhân chính dẫn đến thất bại trong các giao dịch DMA.

4.1 Ba Không gian Địa chỉ Cốt lõi

Hệ thống BCM2711 vận hành trên ba không gian địa chỉ riêng biệt:

1. **Địa chỉ Ảo (Virtual Address - VA):** Được sử dụng bởi các tiến trình user-space và mã kernel (khi truy cập qua con trỏ void*).
2. **Địa chỉ Vật lý ARM (Physical Address - PA):** Địa chỉ mà các lõi CPU ARM nhìn thấy. Trên Pi 4, RAM hệ thống bắt đầu từ 0x00000000.
3. **Địa chỉ Bus (Bus Address - BA):** Địa chỉ mà GPU VideoCore và các ngoại vi DMA (bao gồm Unicam) nhìn thấy.

4.2 Cơ chế Dịch thuật và DMA

BCM2711 tích hợp một đơn vị quản lý bộ nhớ I/O (IOMMU) giản lược hoặc các bảng ánh xạ để chuyển đổi giữa PA và BA.

- Đối với RAM hệ thống (1GB đầu tiên), địa chỉ Bus thường được ánh xạ bắt đầu từ 0xC0000000 (có cache L2) hoặc các alias khác tùy thuộc vào cấu hình.
- **Quy tắc Vàng:** Trình điều khiển **tuyệt đối không được** tự ý chuyển đổi địa chỉ ảo sang vật lý rồi giả định đó là địa chỉ Bus. Thay vào đó, phải sử dụng dma_addr_t được trả về bởi hàm dma_alloc_coherent. Đây là địa chỉ hợp lệ duy nhất mà phần cứng Unicam có thể hiểu được.[8] Việc ghi một địa chỉ vật lý ARM vào thanh ghi IBSA0 sẽ khiến Unicam ghi dữ liệu vào một vùng nhớ sai lệch hoặc gây lỗi bus.

4.3 Cấp phát Bộ nhớ Liên tục (CMA)

Unicam yêu cầu bộ nhớ đích phải liền mạch về mặt vật lý (trong không gian địa chỉ Bus) để thực hiện ghi dữ liệu tuyến tính. Hệ thống Linux trên Raspberry Pi dành riêng một vùng bộ nhớ CMA (được định nghĩa trong cmdline.txt với tham số cma=...) cho mục đích này.[10] Khi gọi dma_alloc_coherent, kernel sẽ tự động trích xuất bộ nhớ từ vùng CMA này, đảm bảo tính liền mạch và tính nhất quán (coherency) giữa CPU và thiết bị DMA.

5. Chi tiết Triển khai Trình điều khiển: Bản đồ Thanh ghi và Cấu hình

Phần này đi sâu vào chi tiết kỹ thuật của việc định nghĩa và thao tác các thanh ghi Unicam, dựa trên các tài liệu kỹ thuật và mã nguồn kernel tham khảo.[11]

5.1 Định nghĩa Header Thanh ghi (bcm2835-unicam-regs.h)

Để điều khiển Unicam thủ công, chúng ta cần một tệp tiêu đề C định nghĩa chính xác các

offset thanh ghi so với địa chỉ cơ sở (0xfe801000).

C

```
#ifndef _BCM2835_UNICAM_REGS_H_
#define _BCM2835_UNICAM_REGS_H_

#include <linux/bits.h>

/* Địa chỉ cơ sở Unicam 1 (CSI1) trên BCM2711 ánh xạ về 0xFE801000 */
/* Các offset dưới đây cộng vào địa chỉ ảo sau khi ioremap */

#define UNICAM_CTRL       0x000   /* Control Register - Điều khiển chính */
#define UNICAM_STA        0x004   /* Status Register - Trạng thái */
#define UNICAM_ANA        0x008   /* Analog Control - Cấu hình PHY */
#define UNICAM_CLK        0x010   /* Clock Control - Quản lý xung nhịp */
#define UNICAM_ICTL       0x100   /* Image Control - Cấu hình ngắt & kích hoạt */
#define UNICAM_ISTA       0x104   /* Image Status - Trạng thái ngắt */
#define UNICAM_IDI0       0x108   /* Image Data Identifier 0 - Định danh dữ liệu CSI */
#define UNICAM_IBSA0      0x110   /* Image Buffer Start Address 0 - Địa chỉ bắt đầu DMA */
#define UNICAM_IBEA0      0x114   /* Image Buffer End Address 0 - Địa chỉ kết thúc DMA */
#define UNICAM_IBWP       0x11c   /* Image Buffer Write Pointer - Con trỏ ghi hiện tại */
#define UNICAM_IPIPE      0x10c   /* Image Pipe Control - Xử lý đường ống dữ liệu */
#define UNICAM_MISC       0x400   /* Miscellaneous Control - Các cấu hình khác */

/* Các Bitmask quan trọng cho UNICAM_CTRL */
#define UNICAM_CPE        BIT(0)  /* Core Peripheral Enable: Bật khối Unicam */
#define UNICAM_HPE        BIT(1)  /* Header Peripheral Enable: Xử lý gói header */

/* Các Bitmask cho UNICAM_ICTL - Quản lý Ngắt */
#define UNICAM_FIE        BIT(0)  /* Frame End Interrupt Enable: Ngắt kết thúc khung */
#define UNICAM_FSIE       BIT(1)  /* Frame Start Interrupt Enable: Ngắt bắt đầu khung */
#define UNICAM_LIE        BIT(2)  /* Line Interrupt Enable: Ngắt dòng (ít dùng) */
#define UNICAM_TFC        BIT(3)  /* Trigger Frame Capture: Kích hoạt chụp */

/* Các Bitmask cho UNICAM_ISTA - Trạng thái Ngắt */
#define UNICAM_IS         BIT(0)  /* Frame End Status: Cờ báo kết thúc khung */
#define UNICAM_FSS        BIT(1)  /* Frame Start Status: Cờ báo bắt đầu khung */
#define UNICAM_LS         BIT(2)  /* Line Status */
#define UNICAM_CRCE       BIT(3)  /* CRC Error: Lỗi toàn vẹn dữ liệu CSI-2 */

/* Cấu hình Định dạng Dữ liệu (IDI0) */
```

```
/* Virtual Channel (VC) = 0, Data Type (DT) tùy thuộc vào cảm biến */
/* RAW10 = 0x2B, RAW8 = 0x2A */
#define UNICAM_IDI0_DT_MASK 0x3F
#define UNICAM_IDI0_VC_MASK 0xC0

#endif /* _BCM2835_UNICAM_REGS_H_ */
```

5.2 Quy trình Khởi tạo và Cấu hình Unicam

Việc khởi tạo Unicam đòi hỏi một trình tự nghiêm ngặt để đảm bảo phần cứng sẵn sàng nhận dữ liệu MIPI tốc độ cao mà không bị treo (lockup) hay tràn bộ đệm FIFO.[12]

1. **Kích hoạt Xung nhịp (Clock Enable):** Trước khi truy cập bất kỳ thanh ghi nào, trình điều khiển phải yêu cầu hệ thống cấp xung nhịp cho VPU và khối Unicam thông qua API clk_prepare_enable. Đối với BCM2711, xung nhịp VPU tối thiểu phải đạt **250MHz** để đảm bảo băng thông xử lý dữ liệu đầu vào mà không gây tràn FIFO.
2. **Thiết lập lại (Reset):** Đảm bảo UNICAM_CPE (bit 0 của UNICAM_CTRL) được xóa về 0 để đưa khối vào trạng thái nghỉ.
3. **Cấu hình Làn dữ liệu (Data Lanes):** Thiết lập số lượng làn dữ liệu hoạt động (2 làn cho IMX219 trên Pi 4) thông qua các thanh ghi UNICAM_ANA hoặc thông qua khối D-PHY tương ứng.
4. **Cấu hình DMA:**
    - Ghi **Địa chỉ Bus** (dma_addr_t) của vùng nhớ CMA đã cấp phát vào UNICAM_IBSA0.
    - Ghi giá trị Địa chỉ Bus + Kích thước Buffer vào UNICAM_IBEA0.
    - *Lưu ý:* Vùng nhớ từ IBSA0 đến IBEA0 sẽ hoạt động như một bộ đệm vòng. Phần cứng Unicam sẽ tự động quay vòng con trỏ ghi (IBWP) khi chạm mốc IBEA0.
5. **Định dạng Dữ liệu (Format Setup):**
    - Ghi mã định danh dữ liệu (Data Type) vào UNICAM_IDI0. Với IMX219 ở chế độ Raw10, giá trị DT là 0x2B. Nếu sử dụng Raw8, giá trị là 0x2A.[13]
    - Cấu hình UNICAM_IPIPE: Quyết định xem dữ liệu có được đóng gói (pack) hay không. Thông thường, Unicam giải nén Raw10 thành 16-bit word trong bộ nhớ để CPU dễ xử lý. Để tối ưu băng thông, có thể cần tắt các chế độ xử lý thừa.
6. **Kích hoạt Ngắt:** Bật UNICAM_FIE (Kết thúc khung) và UNICAM_FSIE (Bắt đầu khung) trong UNICAM_ICTL để CPU nhận biết tiến trình.
7. **Kích hoạt Khối:** Cuối cùng, thiết lập UNICAM_CPE = 1 trong UNICAM_CTRL để bắt đầu lắng nghe dữ liệu trên bus CSI-2.

6. Điều khiển Cảm biến IMX219 qua I2C

Trong khi Unicam chịu trách nhiệm nhận dữ liệu, IMX219 chịu trách nhiệm tạo ra dữ liệu đó. Trình điều khiển phải thực hiện giao tiếp I2C để cấu hình cảm biến.

6.1 Giao tiếp I2C và Multiplexer

Cảm biến IMX219 có địa chỉ I2C 7-bit là 0x10. Trên Raspberry Pi 4, bus I2C0 (hoặc I2C10/I2C_VC tùy cấu hình kernel) được định tuyến qua một bộ dồn kênh (multiplexer) để kết nối với cổng camera. Trình điều khiển cần lấy được struct i2c_client chính xác để gửi lệnh.

6.2 Chuỗi Khởi tạo Chế độ 7 (640x480 High-FPS)

Chế độ này là trọng tâm của yêu cầu. Việc cấu hình sai các thanh ghi binning hoặc PLL sẽ dẫn đến sai lệch tốc độ khung hình hoặc mất tín hiệu. Dưới đây là bảng các giá trị thanh ghi (địa chỉ 16-bit, dữ liệu 8-bit) được tổng hợp từ các nguồn nghiên cứu và phân tích logic analyzer.[7]

Bảng 1: Chuỗi Lệnh I2C cho IMX219 Mode 7 (640x480, ~200fps, 2-lane)

Địa chỉ (Hex)	Giá trị (Hex)	Mô tả Chức năng
0x0100	0x00	**Mode Select**: Standby (Dừng cảm biến để cấu hình)
0x30EB	0x05	Cấu hình truy cập thanh ghi CSI
0x30EB	0x0C	Cấu hình truy cập thanh ghi CSI
0x300A	0xFF	Cấu hình Clock nội bộ
0x300B	0xFF	Cấu hình Clock nội bộ
0x30EB	0x05	Cấu hình truy cập thanh ghi CSI
0x30EB	0x09	Cấu hình truy cập thanh ghi CSI
0x0114	0x01	**CSI Lane Mode**: Thiết lập 2 làn dữ liệu
0x0128	0x00	Điều khiển D-PHY
0x012A	0x18	**EXCK_FREQ**: Tần số xung nhịp ngoại vi (24 MHz)
0x012B	0x00	EXCK_FREQ LSB
0x0157	0x00	Analog Gain Global (Mặc định)
0x0160	0x01	**FRM_LENGTH_A** (MSB): Độ dài khung hình (quy định FPS)
0x0161	0x11	**FRM_LENGTH_A** (LSB): Giá trị càng nhỏ, FPS càng cao
0x0162	0x0D	LINE_LENGTH_A (MSB): Độ dài dòng
0x0163	0xE8	LINE_LENGTH_A (LSB)
0x0164	0x03	X_ADD_STA_A: Tọa độ X bắt đầu cắt (Crop Start)
0x0165	0xE8	X_ADD_STA_A LSB
0x0166	0x08	X_ADD_END_A: Tọa độ X kết thúc cắt

0x0167	0xE7	X_ADD_END_A LSB
0x0168	0x02	Y_ADD_STA_A: Tọa độ Y bắt đầu cắt
0x0169	0xF0	Y_ADD_STA_A LSB
0x016A	0x06	Y_ADD_END_A: Tọa độ Y kết thúc cắt
0x016B	0xAF	Y_ADD_END_A LSB
0x0174	0x03	**BINNING_MODE_H_A**: x2 Analog (Special Binning)
0x0175	0x03	**BINNING_MODE_V_A**: x2 Analog (Special Binning)
0x018C	0x0A	**CSI_DATA_FORMAT_A**: 0x0A = Raw10, 0x08 = Raw8
0x018D	0x0A	CSI_DATA_FORMAT_B
0x0301	0x05	VTPXCK_DIV: Bộ chia xung pixel video
0x0303	0x01	VTSYCK_DIV: Bộ chia xung hệ thống video
0x0304	0x03	PREPLLCK_VT_DIV
0x0305	0x03	PREPLLCK_OP_DIV
0x0306	0x00	**PLL_VT_MPY** (MSB): Hệ số nhân PLL Video Timing
0x0307	0x39	**PLL_VT_MPY** (LSB): Quyết định xung nhịp nội bộ
0x0309	0x0A	OP_PIX_CLK_DIV: Bộ chia xung pixel đầu ra
0x0100	0x01	**Mode Select**: Streaming On (Bắt đầu gửi dữ liệu)

Phân tích Chi tiết:
Các thanh ghi 0x0174 và 0x0175 được thiết lập giá trị 0x03 là yếu tố then chốt. Giá trị này kích hoạt chế độ "Special Analog Binning". Khác với binning kỹ thuật số (0x01), chế độ này giảm thời gian đọc cảm biến, cho phép đạt tốc độ khung hình rất cao mà không bị giới hạn bởi tốc độ chuyển đổi ADC của toàn bộ mảng điểm ảnh.6 Thanh ghi 0x0160/0x0161 (Frame Length) xác định tổng thời gian của một khung hình (bao gồm cả thời gian xóa dọc V-Blank); giảm giá trị này sẽ tăng FPS, nhưng phải đảm bảo không nhỏ hơn thời gian phơi sáng tối thiểu.

7. Xử lý Ngắt và Đồng bộ hóa

Việc bỏ qua videobuf2 đồng nghĩa với việc trình điều khiển phải tự quản lý tín hiệu từ phần cứng.

7.1 Định tuyến Ngắt GIC

Trên BCM2711, bộ điều khiển ngắt GIC-400 quản lý và phân phối ngắt tới các lõi CPU. Các ngắt từ Unicam được định tuyến cố định.

- Phân tích Device Tree cho thấy **Unicam 1 (CSI1)** được gán tới ngắt **GIC_SPI 103**.[15]
- Unicam 0 (CSI0) sử dụng ngắt GIC_SPI 102.

7.2 Chiến lược Xử lý Ngắt (ISR)

Trình điều khiển cần đăng ký hàm xử lý ngắt (request_irq). Do tần suất ngắt rất cao (200Hz+), hàm ISR cần được tối ưu hóa tối đa:

1. **Đọc Trạng thái:** Đọc thanh ghi UNICAM_ISTA để xác định nguyên nhân ngắt.
2. **Xóa Cờ Ngắt:** Ghi lại chính giá trị vừa đọc vào UNICAM_ISTA để xóa cờ (cơ chế write-1-to-clear).
3. **Xử lý Logic:**
    - Nếu là **Frame Start (UNICAM_FSS)**: Ghi nhận thời điểm bắt đầu DMA.
    - Nếu là **Frame End (UNICAM_IS)**: Khung hình đã hoàn tất. Lúc này, con trỏ ghi IBWP đã di chuyển. Trong chế độ bộ đệm vòng, trình điều khiển cần cập nhật các biến trạng thái nội bộ để thông báo cho user-space biết dữ liệu mới đã sẵn sàng.
    - **Cảnh báo:** Nếu xảy ra lỗi CRC (UNICAM_CRCE), cần ghi log hoặc đánh dấu khung hình bị hỏng.

8. Tích hợp Device Tree Overlay

Để ngăn trình điều khiển tiêu chuẩn bcm2835-unicam chiếm dụng phần cứng, cần sử dụng Device Tree Overlay để vô hiệu hóa node mặc định và (tùy chọn) khai báo node tùy chỉnh.
Mã nguồn Overlay mẫu (custom-unicam-overlay.dts):

DTS

```
/dts-v1/;
/plugin/;

/{
    compatible = "brcm,bcm2711";

    /* Vô hiệu hóa driver unicam tiêu chuẩn */
    fragment@0 {
        target = <&csi1>;
        __overlay__ {
            status = "disabled";
        };
    };
```

```
    /* Đảm bảo I2C0/I2C10 được bật để giao tiếp với cảm biến */
    fragment@1 {
        target = <&i2c0>;
        __overlay__ {
            status = "okay";
        };
    };

    /* Định nghĩa node cho driver tùy chỉnh */
    fragment@2 {
        target-path = "/soc";
        __overlay__ {
            my_unicam: my_unicam@7e801000 {
                compatible = "vendor,manual-unicam";
                /* Kernel sẽ tự động ánh xạ 0x7e... sang 0xfe... */
                reg = <0x7e801000 0x800>;
                interrupts = <GIC_SPI 103 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clocks BCM2835_CLOCK_CAM1>,
                         <&firmware_clocks 4>; /* VPU clock */
                clock-names = "lp", "vpu";
            };
        };
    };
};
```

9. Triển khai Mã nguồn Driver (Pseudo-code Chi tiết)

Dưới đây là cấu trúc logic của module kernel:

C

```
/* Module Init/Probe */
static int my_unicam_probe(struct platform_device *pdev) {
    // 1. Map địa chỉ thanh ghi
    struct resource *res = platform_get_resource(pdev, IORESOURCE_MEM, 0);
    priv->reg_base = devm_ioremap_resource(&pdev->dev, res);

    // 2. Lấy thông tin ngắt và đăng ký ISR
    int irq = platform_get_irq(pdev, 0);
    devm_request_irq(&pdev->dev, irq, unicam_isr, 0, "my_unicam", priv);
```

```
    // 3. Cấp phát bộ nhớ DMA (CMA)
    // Cấp phát 16MB liên tục, không cache (coherent)
    priv->buf_size = 16 * 1024 * 1024;
    priv->virt_addr = dma_alloc_coherent(&pdev->dev, priv->buf_size,
                                         &priv->bus_addr, GFP_KERNEL);

    // 4. Tạo char device để user-space giao tiếp
    //...
}

/* Cấu hình Unicam (được gọi từ ioctl hoặc open) */
void configure_hw(struct custom_dev *priv) {
    // Tắt Unicam
    writel(0, priv->reg_base + UNICAM_CTRL);

    // Thiết lập địa chỉ DMA (QUAN TRỌNG: Dùng địa chỉ Bus)
    writel(priv->bus_addr, priv->reg_base + UNICAM_IBSA0);
    writel(priv->bus_addr + priv->buf_size, priv->reg_base + UNICAM_IBEA0);

    // Thiết lập định dạng (Raw10)
    writel(0x2B, priv->reg_base + UNICAM_IDI0);

    // Bật ngắt Frame End
    writel(UNICAM_FIE, priv->reg_base + UNICAM_ICTL);

    // Bật Unicam
    writel(UNICAM_CPE, priv->reg_base + UNICAM_CTRL);
}

/* User-space mmap handler */
static int my_mmap(struct file *filp, struct vm_area_struct *vma) {
    // Ánh xạ vùng nhớ DMA coherent vào user-space
    return dma_mmap_coherent(priv->dev, vma, priv->virt_addr,
                             priv->bus_addr, priv->buf_size);
}
```

10. Kết luận và Kiến nghị Chiến lược

Việc triển khai trình điều khiển DMA thủ công cho BCM2711 Unicam là một giải pháp kỹ thuật phức tạp nhưng mang lại hiệu quả vượt trội cho các ứng dụng thị giác máy tính thời gian thực. Bằng cách loại bỏ các lớp trung gian của videobuf2, hệ thống đạt được độ trễ truyền tải thấp

nhất có thể (chỉ phụ thuộc vào tốc độ đọc cảm biến và băng thông MIPI), đồng thời kiểm soát hoàn toàn việc quản lý bộ nhớ.

Các điểm mấu chốt cần lưu ý:

1. **Tính Nhất quán Bộ nhớ (Coherency):** Sử dụng dma_alloc_coherent là bắt buộc để đảm bảo địa chỉ Bus chính xác và dữ liệu được đồng bộ giữa Unicam và CPU.
2. **Cấu hình Cảm biến:** Chế độ Analog Binning (Mode 7) của IMX219 phải được cấu hình chính xác qua I2C để khớp với băng thông và định dạng mà Unicam mong đợi.
3. **Xử lý Ngắt:** ISR phải cực kỳ gọn nhẹ để tránh làm nghẽn CPU khi hoạt động ở tốc độ 200fps+.

Giải pháp này cung cấp nền tảng vững chắc cho các ứng dụng như robot tự hành, hệ thống kiểm tra công nghiệp tốc độ cao, và các tác vụ AI edge đòi hỏi luồng dữ liệu thô (raw data) liên tục và ổn định.

Nguồn trích dẫn

1. BCM2711 ARM Peripherals - Raspberry Pi Datasheets, truy cập vào tháng 1 5, 2026, https://datasheets.raspberrypi.com/bcm2711/bcm2711-peripherals.pdf
2. Rapberry pi 4 IO memory base address - Raspberry Pi Forums, truy cập vào tháng 1 5, 2026, https://forums.raspberrypi.com/viewtopic.php?t=244031
3. How to configure unicam to enable CSI2's Virtual Channel 1 - Raspberry Pi Forums, truy cập vào tháng 1 5, 2026, https://forums.raspberrypi.com/viewtopic.php?t=382871
4. ebf-buster-linux/Documentation/devicetree/bindings/media/bcm2835-unicam.txt at master, truy cập vào tháng 1 5, 2026, https://github.com/Embedfire/ebf-buster-linux/blob/master/Documentation/devicetree/bindings/media/bcm2835-unicam.txt
5. Bug #1926755 “Support Auvidea B101 (TC358743)” : Bugs : linux, truy cập vào tháng 1 5, 2026, https://bugs.launchpad.net/bugs/1926755
6. [imx219] 10bit 640x480 video at high fps (200+) · Issue #5493 · raspberrypi/linux - GitHub, truy cập vào tháng 1 5, 2026, https://github.com/raspberrypi/linux/issues/5493
7. v2 camera at 640x480 runs only up to 100fps with v4l2 - Raspberry Pi Forums, truy cập vào tháng 1 5, 2026, https://forums.raspberrypi.com/viewtopic.php?t=300105
8. From where does dma_alloc_coherent() attempt to get memory? - Processors forum - TI E2E, truy cập vào tháng 1 5, 2026, https://e2e.ti.com/support/processors-group/processors/f/processors-forum/89419/from-where-does-dma_alloc_coherent-attempt-to-get-memory
9. Dynamic DMA mapping using the generic device - The Linux Kernel documentation, truy cập vào tháng 1 5, 2026, https://docs.kernel.org/core-api/dma-api.html
10. A deep dive into CMA - LWN.net, truy cập vào tháng 1 5, 2026, https://lwn.net/Articles/486301/
11. drivers/media/platform/broadcom/bcm2835-unicam-regs.h - pub/scm/linux/kernel/git/remoteproc/linux - Git at Google, truy cập vào tháng 1 5,

2026, https://kernel.googlesource.com/pub/scm/linux/kernel/git/remoteproc/linux/+/refs/tags/rpmsg-v6.13/drivers/media/platform/broadcom/bcm2835-unicam-regs.h
12. drivers/media/platform/broadcom/bcm2835-unicam.c - kernel/common - Git at Google, truy cập vào tháng 1 5, 2026, https://android.1.googlesource.com/kernel/common/+/24172e0d79900908cf5ebf366600616d29c9b417/drivers/media/platform/broadcom/bcm2835-unicam.c
13. Raspiraw + Raspi3B + V2 camera - 8 bit packing gives black level offset and overdrivwen white, truy cập vào tháng 1 5, 2026, https://forums.raspberrypi.com/viewtopic.php?t=281165
14. [BUG] imx219 binned 2x 8bits mode produces corrupted frames · Issue #281 · raspberrypi/rpicam-apps - GitHub, truy cập vào tháng 1 5, 2026, https://github.com/raspberrypi/rpicam-apps/issues/281
15. [RFC PATCH v2 5/7] ARM: dts: bcm2711: Add unicam CSI nodes - Mailing Lists, truy cập vào tháng 1 5, 2026, https://lists.infradead.org/pipermail/linux-arm-kernel/2022-January/711418.html
16. brcm,bcm2835-unicam.yaml, truy cập vào tháng 1 5, 2026, https://www.kernel.org/doc/Documentation/devicetree/bindings/media/brcm%2Cbcm2835-unicam.yaml