FreeBSD on Freescale QorIQ Data Path Acceleration Architecture Devices

Piotr Zięcik
kosmo@semihalf.com

BSDCan 2012 9-13 May, 2012
Presentation outline

- Introduction,
FreeBSD on Freescale QorIQ Data Path Acceleration Architecture Devices

Presentation outline

- Introduction,
- Hardware description:
Presentation outline

- Introduction,
- Hardware description:
  - Data Path Acceleration Architecture.
Presentation outline

- Introduction,
- Hardware description:
  - Data Path Acceleration Architecture.
- Software description:
Presentation outline

- Introduction,
- Hardware description:
  - Data Path Acceleration Architecture.
- Software description:
  - Porting process.
Presentation outline

- Introduction,
- Hardware description:
  - Data Path Acceleration Architecture.
- Software description:
  - Porting process.
- Current state,
Presentation outline

- Introduction,
- Hardware description:
  - Data Path Acceleration Architecture.
- Software description:
  - Porting process.
- Current state,
- Future work.
Introduction

Networks are too fast!
Introduction

Networks are too fast!

- High packet rate:
Introduction

Networks are too fast!

- High packet rate:
  - Fast Ethernet: 182 kpps.
Introduction

Networks are too fast!

- High packet rate:
  - Fast Ethernet: 182 kpps.
  - Gigabit Ethernet: 1.46 Mpps
Introduction

Networks are too fast!

- High packet rate:
  - Fast Ethernet: 182 kpps.
  - Gigabit Ethernet: 1.46 Mpps (4k ring buffer full in 2.8 ms).
Introduction

Networks are too fast!

- High packet rate:
  - Fast Ethernet: 182 kpps.
  - Gigabit Ethernet: 1.46 Mpps (4k ring buffer full in 2.8 ms).

- Problems:
Introduction

Networks are too fast!

- High packet rate:
  - Fast Ethernet: 182 kpps.
  - Gigabit Ethernet: 1.46 Mpps (4k ring buffer full in 2.8 ms).

- Problems:
  - High interrupt rate and/or latency,
Introduction

Networks are too fast!

- High packet rate:
  - Fast Ethernet: 182 kpps.
  - Gigabit Ethernet: 1.46 Mpps (4k ring buffer full in 2.8 ms).

- Problems:
  - High interrupt rate and/or latency,
  - DMA memory management overhead,
Introduction

Networks are too fast!

- High packet rate:
  - Fast Ethernet: 182 kpps.
  - Gigabit Ethernet: 1.46 Mpps (4k ring buffer full in 2.8 ms).

- Problems:
  - High interrupt rate and/or latency,
  - DMA memory management overhead,
  - Lock congestion,
Introduction

Networks are too fast!

- High packet rate:
  - Fast Ethernet: 182 kpps.
  - Gigabit Ethernet: 1.46 Mpps (4k ring buffer full in 2.8 ms).

- Problems:
  - High interrupt rate and/or latency,
  - DMA memory management overhead,
  - Lock congestion,
  - Packet parsing/inspection overhead.
Freescale QoriQ DPAA Family

- Most recent Freescale Communication SoCs,
Freescale QoriQ DPAA Family

- Most recent Freescale Communication SoCs,
- Successor of the PowerQUICC family,
Freescale QoriQ DPAA Family

- Most recent Freescale Communication SoCs,
- Successor of the PowerQUICC family,
- Up to 8 e500mc Cores,
Freescale QoriQ DPAA Family

- Most recent Freescale Communication SoCs,
- Successor of the PowerQUICC family,
- Up to 8 e500mc Cores,
- Up to 2x 10GbE + 8x 1GbE,
Freescale QorIQ DPAA Family

- Most recent Freescale Communication SoCs,
- Successor of the PowerQUICC family,
- Up to 8 e500mc Cores,
- Up to 2x 10GbE + 8x 1GbE,
- Hardware Virtualization Support,
Freescale QoriQ DPAA Family

- Most recent Freescale Communication SoCs,
- Successor of the PowerQUICC family,
- Up to 8 e500mc Cores,
- Up to 2x 10GbE + 8x 1GbE,
- Hardware Virtualization Support,
- Data Path Acceleration Architecture.
FreeBSD on Freescale QorIQ Data Path Acceleration Architecture Devices

Freescale QorIQ DPAA Family

128 KB Backside L2 Cache
Power Architecture® e500-mc Core
32 KB D-Cache
32 KB I-Cache
1024 KB Frontside CoreNet Platform Cache
64-bit DDR3/3L Memory Controller

Security Fuse Processor
Security Monitor
2x USB 2.0 with PHY
eSDHC
eLBC
SD/MMC
2x DUART
2x I²C
SPI, GPIO

CoreNet Coherency Fabric
PAMU
PAMU
PAMU
PAMU
Peripheral Access Management Unit

Frame Manager
Parse, Classify, Distribute
SATA 2.0
SATA 2.0
DMA
DMA
PCle
PCle
PCle
SRIO
SRIO

Pattern Matching Engine 2.1
Queue Manager
Buffer Manager

18-Lane 5 GHz SerDes

QorIQ P3041 Communication Processor (source: P3041 Fact Sheet)
Data Path Acceleration Architecture

Components
Data Path Acceleration Architecture

Components

- Buffer Manager,
- Queue Manager,
- Frame Manager (NICs),
- Others:
  - Security Accelerator,
  - Pattern Matching Engine.
Data Path Acceleration Architecture

Components

- Buffer Manager,
- Queue Manager,
Data Path Acceleration Architecture

Components

▶ Buffer Manager,
▶ Queue Manager,
▶ Frame Manager (NICs),
Data Path Acceleration Architecture

Components

- Buffer Manager,
- Queue Manager,
- Frame Manager (NICs),
- Others:
  - Security Accelerator,
  - Pattern Matching Engine.
Data Path Acceleration Architecture

Components

- Buffer Manager,
- Queue Manager,
- Frame Manager (NICs),
- Others:
  - Security Accelerator,
Data Path Acceleration Architecture

Components

- Buffer Manager,
- Queue Manager,
- Frame Manager (NICs),
- Others:
  - Security Accelerator,
  - Pattern Matching Engine.
Data Path Acceleration Architecture

Components: Buffer Manager
Data Path Acceleration Architecture

Components: Buffer Manager

- Maintains Buffers,
Data Path Acceleration Architecture

Components: Buffer Manager

- Maintains Buffers,
- Pools = Set of Buffers,
Data Path Acceleration Architecture

Components: Buffer Manager

- Maintains Buffers,
- Pools = Set of Buffers,
- Pools are managed by software,
Data Path Acceleration Architecture

Components: Buffer Manager

- Maintains Buffers,
- Pools = Set of Buffers,
- Pools are managed by software,
- Software and hardware may directly allocate/free buffer.
Data Path Acceleration Architecture

Components: Queue Manager
Data Path Acceleration Architecture

Components: Queue Manager

- Maintains Frames,
Data Path Acceleration Architecture

Components: Queue Manager

- Maintains Frames,
- Frame Queue = Linked list of Frames,
Data Path Acceleration Architecture

Components: Queue Manager

- Maintains Frames,
- Frame Queue = Linked list of Frames,
- Work Queue = Linked list of Frame Queues,
Components: Queue Manager

- Maintains Frames,
- Frame Queue = Linked list of Frames,
- Work Queue = Linked list of Frame Queues,
- Channel = 8 * (Work Queue + Priority).
Data Path Acceleration Architecture

Components: Queue Manager

- Frames are enqueued to Frame Queues,
Data Path Acceleration Architecture

Components: Queue Manager

- Frames are enqueued to Frame Queues,
- Frames are dequeued from Channels,
Data Path Acceleration Architecture

Components: Queue Manager

- Frames are enqueued to Frame Queues,
- Frames are dequeued from Channels,
- Channels are attached to SoC components:
Data Path Acceleration Architecture

Components: Queue Manager

- Frames are enqueued to Frame Queues,
- Frames are dequeued from Channels,
- Channels are attached to SoC components:
  - Dedicated Channel: Connected to single device,
Data Path Acceleration Architecture

Components: Queue Manager

- Frames are enqueued to Frame Queues,
- Frames are dequeued from Channels,
- Channels are attached to SoC components:
  - Dedicated Channel: Connected to single device,
  - Poll Channel: Connected to group of devices.
Data Path Acceleration Architecture

Components: Queue Manager

- Frames are enqueued to Frame Queues,
- Frames are dequeued from Channels,
- Channels are attached to SoC components:
  - Dedicated Channel: Connected to single device,
  - Poll Channel: Connected to group of devices.
- Other Queue Manager features:
Data Path Acceleration Architecture

Components: Queue Manager

- Frames are enqueued to Frame Queues,
- Frames are dequeued from Channels,
- Channels are attached to SoC components:
  - Dedicated Channel: Connected to single device,
  - Poll Channel: Connected to group of devices.

- Other Queue Manager features:
  - Congestion Management (Tail drop, RED/WRED),
Data Path Acceleration Architecture

Components: Queue Manager

- Frames are enqueued to Frame Queues,
- Frames are dequeued from Channels,
- Channels are attached to SoC components:
  - Dedicated Channel: Connected to single device,
  - Poll Channel: Connected to group of devices.
- Other Queue Manager features:
  - Congestion Management (Tail drop, RED/WRED),
  - Frame order restoration.
Data Path Acceleration Architecture

Components: Frame Manager

MACs: 1x 10Gb Ethernet + 5x 1Gb Ethernet.

DMA:
- Buffer Manager Interface (QMI).
- Queue Manager Interface (BMI), DMA Engine,
- Frame Processor:
  - Parser,
  - Key Generator,
  - Policer,
  - Frame Processing Module.
Data Path Acceleration Architecture

Components: Frame Manager

- MACs:
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.

- **DMA:**
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.

- **DMA:**
  - Buffer Manager Interface (QMI).
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.

- **DMA:**
  - Buffer Manager Interface (QMI).
  - Queue Manager Interface (BMI),
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.

- **DMA:**
  - Buffer Manager Interface (QMI).
  - Queue Manager Interface (BMI).
  - DMA Engine,
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.

- **DMA:**
  - Buffer Manager Interface (QMI).
  - Queue Manager Interface (BMI),
  - DMA Engine,

- **Frame Processor:**
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.

- **DMA:**
  - Buffer Manager Interface (QMI).
  - Queue Manager Interface (BMI),
  - DMA Engine,

- **Frame Processor:**
  - Parser,
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.

- **DMA:**
  - Buffer Manager Interface (QMI).
  - Queue Manager Interface (BMI).
  - DMA Engine.

- **Frame Processor:**
  - Parser,
  - Key Generator,
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.

- **DMA:**
  - Buffer Manager Interface (QMI).
  - Queue Manager Interface (BMI),
  - DMA Engine,

- **Frame Processor:**
  - Parser,
  - Key Generator,
  - Policer,
Data Path Acceleration Architecture

Components: Frame Manager

- **MACs:**
  - 1x 10Gb Ethernet + 5x 1Gb Ethernet.

- **DMA:**
  - Buffer Manager Interface (QMI).
  - Queue Manager Interface (BMI),
  - DMA Engine,

- **Frame Processor:**
  - Parser,
  - Key Generator,
  - Policer,
  - Frame Processing Module.
Data Path Acceleration Architecture

Software Portals
Data Path Acceleration Architecture

Software Portals

- CPU ↔ DPAA communication channels,
Software Portals

- CPU ↔ DPAA communication channels,
- Portal = Cache Enabled + Cache Inhibited registers,
Data Path Acceleration Architecture

Software Portals

- CPU ↔ DPAA communication channels,
- Portal = Cache Enabled + Cache Inhibited registers,
- Transaction oriented,
Data Path Acceleration Architecture

Software Portals

- CPU ↔ DPAA communication channels,
- Portal = Cache Enabled + Cache Inhibited registers,
- Transaction oriented,
- Efficient SoC Bus usage.
Hardware: Other selected features
Hardware: Other selected features

▶ Cache Stashing,
Hardware: Other selected features

- Cache Stashing,
- Virtualization:
Hardware: Other selected features

- Cache Stashing,
- Virtualization:
  - Additional privilege level in e500mc core,
Hardware: Other selected features

- Cache Stashing,
- Virtualization:
  - Additional privilege level in e500mc core,
  - Peripheral Access Management Units,
Hardware: Other selected features

- Cache Stashing,
- Virtualization:
  - Additional privilege level in e500mc core,
  - Peripheral Access Management Units,
    - Logical I/O Device Number,
Hardware: Other selected features

- Cache Stashing,
- Virtualization:
  - Additional privilege level in e500mc core,
  - Peripheral Access Management Units,
    - Logical I/O Device Number,
  - Data Path Acceleration Architecture.
Software
Software

- Toolchain,
Software

- Toolchain,
- Early kernel initialization,
Software

- Toolchain,
- Early kernel initialization,
- Data Path Acceleration Architecture bring-up,
Software

- Toolchain,
- Early kernel initialization,
- Data Path Acceleration Architecture bring-up,
- SMP,
  - DPAA in SMP environment,
Software

- Toolchain,
- Early kernel initialization,
- Data Path Acceleration Architecture bring-up,
- SMP,
  - DPAA in SMP environment,
- Other peripherals.
Toolchain

Support for e500v2 (predecessor of the e500mc) had been already present,
only minor changes were required in binutils and gcc,
all patches were available from the community and Freescale.
Support for e500v2 (predecessor of the e500mc) had been already present,
Toolchain

- Support for e500v2 (predecessor of the e500mc) had been already present,
- Only minor changes were required in binutils and gcc,
Toolchain

- Support for e500v2 (predecessor of the e500mc) had been already present,
- Only minor changes were required in binutils and gcc,
- All patches were available from the community and Freescale.
Early kernel initialization

locore.S
Early kernel initialization

locore.S

- First code executed in FreeBSD kernel,
Early kernel initialization

locore.S

- First code executed in FreeBSD kernel,
- Architecture depended assembly,
Early kernel initialization

locore.S

- First code executed in FreeBSD kernel,
- Architecture depended assembly,
- Prepares environment for C.
Early kernel initialization

locale.S

- First code executed in FreeBSD kernel,
- Architecture depended assembly,
- Prepares environment for C.

PowerPC locale.S
Early kernel initialization

**locore.S**

- First code executed in FreeBSD kernel,
- Architecture depended assembly,
- Prepares environment for C.

**PowerPC locore.S**

- TLB initialization,
Early kernel initialization

**locore.S**
- First code executed in FreeBSD kernel,
- Architecture depended assembly,
- Prepares environment for C.

**PowerPC locore.S**
- TLB initialization,
- Kernel stack initialization.
Early kernel initialization

e500mc features in locore.S
Early kernel initialization

e500mc features in locore.S

- Bigger TLB,
Early kernel initialization

e500mc features in locore.S

- Bigger TLB,
- Hypervisor privilege level,
Early kernel initialization

*e500mc features in locore.S*

- Bigger TLB,
- Hypervisor privilege level,
- Hardware Implementation-Dependent Registers (HIDs).
Early kernel initialization

**e500mc features in locore.S**

- Bigger TLB,
- Hypervisor privilege level,
- Hardware Implementation-Dependent Registers (HIDs).

New TLB also affects `pmap(9)`.
DPAA Bring-up

A packet processing framework,
OS Agnostic,
Greatly reduces development time,
Propetary licensed.

Thanks to Freescale!

Buffer, Queue and Frame Manager drivers are now available under the BSD license.
DPAA Bring-up

NetCommSw
DPAA Bring-up

NetCommSw

- A packet processing framework,
NetCommSw

- A packet processing framework,
- OS Agnostic,
NetCommSw

- A packet processing framework,
- OS Agnostic,
- Greatly reduces development time,
DPAA Bring-up

NetCommSw

- A packet processing framework,
- OS Agnostic,
- Greatly reduces development time,
- Proprietary licensed.

Thanks to Freescale!

Buffer, Queue and Frame Manager drivers are now available under the BSD license.
DPAA Bring-up

NetCommSw

- A packet processing framework,
- OS Agnostic,
- Greatly reduces development time,
- Proprietary licensed.

Thanks to Freescale!
Buffer, Queue and Frame Manager drivers are now available under the BSD license.
NetCommSw Driver Model

- Application
- OS Device Interface
- Wrapper Driver
- NetCommSW Low-level Device Driver
- XX Routines
- OS Low-level Services
- Hardware
XX Routines

Simple routines for basic OS functions:

- `void *XX_Malloc(unsigned int size)`

Physical to Virtual translation is required:

- Not supported by the FreeBSD kernel,
- Ambiguous,

Solution:

- Keep list of all active mappings in `vm_page` structure,
- `pmap_enter()` and `pmap_remove()` manages the list,

XX Routines: PA $\rightarrow$ `vm_page` $\rightarrow$ the mappings list $\rightarrow$ VA.
XX Routines

- Simple routines for basic OS functions:
  - `void *XX_Malloc(unsigned int size)`
  - Physical to Virtual translation is required:
    - Not supported by the FreeBSD kernel,
    - Ambiguous,
  - Solution:
    - Keep list of all active mappings in `vm_page` structure,
    - `pmap_enter()` and `pmap_remove()` manage the list,
  - XX Routines: PA → `vm_page` → the mappings list → VA.
**XX Routines**

- Simple routines for basic OS functions:
  - `void *XXMalloc(unsigned int size)`
**XX Routines**

- Simple routines for basic OS functions:
  - `void *XX_Malloc(unsigned int size)`
- Physical to Virtual translation is required:
  - Solution:
    - Keep list of all active mappings in `vm_page` structure.
    - `pmap_enter()` and `pmap_remove()` manages the list.
  - `XX Routines: PA → vm_page → the mappings list → VA.`
XX Routines

- Simple routines for basic OS functions:
  - `void *XX_Malloc(unsigned int size)`

- Physical to Virtual translation is required:
  - Not supported by the FreeBSD kernel,

- Solution:
  - Keep list of all active mappings in `vm_page` structure,
  - `pmap_enter()` and `pmap_remove()` manages the list,
  - PA → `vm_page` → the mappings list → VA.
XX Routines

- Simple routines for basic OS functions:
  - `void *XX_Malloc(unsigned int size)`

- Physical to Virtual translation is required:
  - Not supported by the FreeBSD kernel,
  - Ambiguous,

- Solution:
  - Keep list of all active mappings in `vm_page` structure,
  - `pmap_enter()` and `pmap_remove()` manages the list,
- XX Routines: PA → `vm_page` → the mappings list → VA.
XX Routines

- Simple routines for basic OS functions:
  - `void *XXMalloc(unsigned int size)`

- Physical to Virtual translation is required:
  - Not supported by the FreeBSD kernel,
  - Ambiguous,

- Solution:
XX Routines

- Simple routines for basic OS functions:
  - void *XX_Malloc(unsigned int size)

- Physical to Virtual translation is required:
  - Not supported by the FreeBSD kernel,
  - Ambiguous,

- Solution:
  - Keep list of all active mappings in vm_page structure,
**XX Routines**

- Simple routines for basic OS functions:
  - `void *XXMalloc(unsigned int size)`

- Physical to Virtual translation is required:
  - Not supported by the FreeBSD kernel,
  - Ambiguous,

- Solution:
  - Keep list of all active mappings in `vm_page` structure,
  - `pmap_enter()` and `pmap_remove()` manages the list,
**XX Routines**

- Simple routines for basic OS functions:
  - `void *XX_Malloc(unsigned int size)`

- Physical to Virtual translation is required:
  - Not supported by the FreeBSD kernel,
  - Ambiguous,

- Solution:
  - Keep list of all active mappings in `vm_page` structure,
  - `pmap_enter()` and `pmap_remove()` manages the list,
  - XX Routines: PA $\rightarrow$ `vm_page` $\rightarrow$ the mappings list $\rightarrow$ VA.
Wrapper Drivers
Wrapper Drivers

- OS ↔ NetCommSw API translation layer,
Wrapper Drivers

- OS ↔ NetCommSw API translation layer,
- Access serialization layer,
Wrapper Drivers

- OS ↔ NetCommSw API translation layer,
- Access serialization layer,
- Written as simple newbus attachments,
Wrapper Drivers

- OS ↔ NetCommSw API translation layer,
- Access serialization layer,
- Written as simple newbus attachments,
- Export own API.
Frame Manager Wrapper Driver

- Consists of several NetCommSw submodules,
- Each submodule have own API,
- A single Frame Manager Wrapper Driver:
  - Performs initialization of common parts,
  - Manages internal FMan resources,
  - Exports single, simplified API.
Frame Manager Wrapper Driver

- Consists of several NetCommSw submodules,
Frame Manager Wrapper Driver

- Consists of several NetCommSw submodules,
- Each submodule have own API,
Frame Manager Wrapper Driver

- Consists of several NetCommSw submodules,
- Each submodule have own API,
- A single Frame Manager Wrapper Driver:
Frame Manager Wrapper Driver

- Consists of several NetCommSw submodules,
- Each submodule have own API,
- A single Frame Manager Wrapper Driver:
  - Performs initialization of common parts,
Frame Manager Wrapper Driver

- Consists of several NetCommSw submodules,
- Each submodule have own API,
- A single Frame Manager Wrapper Driver:
  - Performs initialization of common parts,
  - Manages internal FMan resources,
Frame Manager Wrapper Driver

- Consists of several NetCommSw submodules,
- Each submodule have own API,
- A single Frame Manager Wrapper Driver:
  - Performs initialization of common parts,
  - Manages internal FMan resources,
  - Exports single, simplified API.
dTSEC Driver

- Classic NIC driver from OS perspective,
- Uses BMan, QMan and FMan wrapper drivers,
- Binds DPAA parts together.
dTSEC Driver

- Classic NIC driver from OS perspective,
dTSEC Driver

- Classic NIC driver from OS perspective,
- Uses BMan, QMan and FMan wrapper drivers,
dTSEC Driver

- Classic NIC driver from OS perspective,
- Uses BMan, QMan and FMan wrapper drivers,
- Binds DPAA parts together.
dTSEC Driver: DPAA Usage

- **Buffer Manager:** RX buffer pool management.
- **Queue Manager:** Reads/Writes frames from/to queues (associated with MACs).
- **Frame Manager:** MAC abstraction, data flow configuration.
dTSEC Driver: DPAA Usage

- Buffer Manager:
dTSEC Driver: DPAA Usage

- Buffer Manager:
  - RX buffer pool management.
dTSEC Driver: DPAA Usage

▶ Buffer Manager:
  ▶ RX buffer pool management.

▶ Queue Manager:
dTSEC Driver: DPAA Usage

- **Buffer Manager:**
  - RX buffer pool management.

- **Queue Manager:**
  - Reads/Writes frames from/to queues (associated with MACs).
dTSEC Driver: DPAA Usage

- **Buffer Manager:**
  - RX buffer pool management.

- **Queue Manager:**
  - Reads/Writes frames from/to queues (associated with MACs).

- **Frame Manager:**
Buffer Manager:

- RX buffer pool management.

Queue Manager:

- Reads/Writes frames from/to queues (associated with MACs).

Frame Manager:

- MAC abstraction,
dTSEC Driver: DPAA Usage

- **Buffer Manager:**
  - RX buffer pool management.

- **Queue Manager:**
  - Reads/Writes frames from/to queues (associated with MACs).

- **Frame Manager:**
  - MAC abstraction,
  - Data flow configuration.
dTSEC Driver: TX Path
dTSEC Driver: RX Path

FreeBSD on Freescale QorIQ Data Path Acceleration Architecture Devices
SMP Bring-Up

Based on existing e500v2 implementation,

Issues with more than 2 cores:

- IPI Multicasting
- DPAA-related issues.
SMP Bring-Up

- Based on existing e500v2 implementation,
SMP Bring-Up

- Based on existing e500v2 implementation,
- Issues with more than 2 cores:
SMP Bring-Up

- Based on existing e500v2 implementation,
- Issues with more than 2 cores:
  - IPI Multicasting
SMP Bring-Up

- Based on existing e500v2 implementation,
- Issues with more than 2 cores:
  - IPI Multicasting
- DPAA-related issues.
SMP vs DPAA: Issues

Portal Mapping
SMP vs DPAA: Issues

Portal Mapping

- Dedicated portal for each CPU,
SMP vs DPAA: Issues

Portal Mapping

- Dedicated portal for each CPU,
- Same portal address on each CPU,
SMP vs DPAA: Issues

Portal Mapping

- Dedicated portal for each CPU,
- Same portal address on each CPU,
- Such device mapping was not supported:
SMP vs DPAA: Issues

Portal Mapping

- Dedicated portal for each CPU,
- Same portal address on each CPU,
- Such device mapping was not supported:
  - Introduction of "shared" bit,
SMP vs DPAA: Issues

Portal Mapping

- Dedicated portal for each CPU,
- Same portal address on each CPU,
- Such device mapping was not supported:
  - Introduction of "shared" bit,
  - Implemented using user-defined bits in TLB entry,
SMP vs DPAA: Issues

Portal Mapping

- Dedicated portal for each CPU,
- Same portal address on each CPU,
- Such device mapping was not supported:
  - Introduction of "shared" bit,
  - Implemented using user-defined bits in TLB entry,
  - Only marked entries are propagated to other cores,
SMP vs DPAA: Issues

Portal Mapping

- Dedicated portal for each CPU,
- Same portal address on each CPU,
- Such device mapping was not supported:
  - Introduction of "shared" bit,
  - Implemented using user-defined bits in TLB entry,
  - Only marked entries are propagated to other cores,
  - Private are set on each CPU.
SMP vs DPAA: Issues

Portal Configuration

- Has to be performed on each portal (CPU),
- Only boot CPU can execute configuration during boot,
- Other cores must initialize portals on demand:
  - This may happen any time,
  - Portal configuration includes IRQ request,
  - But `intr_event_create()` may sleep!
  - Solution: Interrupt preallocation in XX Routines.
SMP vs DPAA: Issues

Portal Configuration

- Has to be performed on each portal (CPU),

However, intr_event_create() may sleep!

Solution: Interrupt preallocation in XX Routines.
SMP vs DPAA: Issues

Portal Configuration

- Has to be performed on each portal (CPU),
- Only boot CPU can execute configuration during boot,
SMP vs DPAA: Issues

Portal Configuration

- Has to be performed on each portal (CPU),
- Only boot CPU can execute configuration during boot,
- Other cores must initialize portals on demand:
  - This may happen any time,
  - Portal configuration includes IRQ request,
  - But `intr_event_create()` may sleep!
  - Solution: Interrupt preallocation in XX Routines.
**SMP vs DPAA: Issues**

**Portal Configuration**

- Has to be performed on each portal (CPU),
- Only boot CPU can execute configuration during boot,
- Other cores must initialize portals on demand:
  - This may happen any time,
SMP vs DPAA: Issues

Portal Configuration

- Has to be performed on each portal (CPU),
- Only boot CPU can execute configuration during boot,
- Other cores must initialize portals on demand:
  - This may happen any time,
  - Portal configuration includes IRQ request,
Portal Configuration

- Has to be performed on each portal (CPU),
- Only boot CPU can execute configuration during boot,
- Other cores must initialize portals on demand:
  - This may happen any time,
  - Portal configuration includes IRQ request,
  - But `intr_event_create()` may sleep!
SMP vs DPAA: Issues

Portal Configuration

- Has to be performed on each portal (CPU),
- Only boot CPU can execute configuration during boot,
- Other cores must initialize portals on demand:
  - This may happen any time,
  - Portal configuration includes IRQ request,
  - But `intr_event_create()` may sleep!
  - Solution: Interrupt preallocation in XX Routines.
SMP vs DPAA: Issues

Portal Interrupts
SMP vs DPAA: Issues

Portal Interrupts

- Portal Interrupts must be bound to particular CPU,
SMP vs DPAA: Issues

Portal Interrupts

- Portal Interrupts must be bound to particular CPU,
- Solution: Interrupt thread binding layer in XX Routines.
Other Peripherals

- PCI Express Bus,
- USB Controller (EHCI compliant),
- SD/MMC Controller,
- I2C Controller,
- IDMA Controller.
Other Peripherals

- PCI Express Bus,
Other Peripherals

- PCI Express Bus,
- USB Controller (EHCI compliant),
Other Peripherals

- PCI Express Bus,
- USB Controller (EHCI compliant),
- SD/MMC Controller,
Other Peripherals

- PCI Express Bus,
- USB Controller (EHCI compliant),
- SD/MMC Controller,
- I2C Controller,
Other Peripherals

- PCI Express Bus,
- USB Controller (EHCI compliant),
- SD/MMC Controller,
- I2C Controller,
- IDMA Controller.
Current State and Results

- Supported chips: P2041, P3041 and P5020 (in 32-bit mode)
- Simple iperf test:
  - Speed: 897 Mbit/s
  - CPU Utilization: 30%, Of which 90% was interrupt servicing.
Current State and Results

- Supported chips: P2041, P3041 and P5020 (in 32-bit mode),
- Simple `iperf` test:
  - Speed: 897 Mbit/s
  - CPU Utilization: 30%, Of which 90% was interrupt servicing.
Current State and Results

- Supported chips: P2041, P3041 and P5020 (in 32-bit mode),
- Simple iperf test:
  - Speed: 897 Mbit/s
  - CPU Utilization: 30%, Of which 90% was interrupt servicing.
Current State and Results

- Supported chips: P2041, P3041 and P5020 (in 32-bit mode),
- Simple iperf test:
  - Speed: 897 Mbit/s
Current State and Results

- Supported chips: P2041, P3041 and P5020 (in 32-bit mode),
- Simple `iperf` test:
  - Speed: 897 Mbit/s
  - CPU Utilization: 30%,
Current State and Results

- Supported chips: P2041, P3041 and P5020 (in 32-bit mode),
- Simple `iperf` test:
  - Speed: 897 Mbit/s
  - CPU Utilization: 30%,
  - Of which 90% was interrupt servicing.
### Current State and Results

```bash
p3041# vmstat -i

<table>
<thead>
<tr>
<th>interrupt</th>
<th>total</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>irq121: bman0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>irq120: qman0</td>
<td>2784930</td>
<td>3584</td>
</tr>
<tr>
<td>irq122: qman0</td>
<td>2194130</td>
<td>2823</td>
</tr>
<tr>
<td>irq124: qman0</td>
<td>2263079</td>
<td>2912</td>
</tr>
<tr>
<td>irq126: qman0</td>
<td>2148167</td>
<td>2764</td>
</tr>
</tbody>
</table>
(...)```

Future work

- More networking features:
  - Pooling Mode,
  - Hardware Checksumming,
  - Jumbo Frames,
- More peripherals:
  - Pattern Matching Engine,
  - Security Engine,
  - SATA Controller.
Future work

- More networking features:
Future work

- More networking features:
  - Pooling Mode,
Future work

- More networking features:
  - Pooling Mode,
  - Hardware Checksumming,
Future work

- More networking features:
  - Pooling Mode,
  - Hardware Checksumming,
  - Jumbo Frames,
Future work

- More networking features:
  - Pooling Mode,
  - Hardware Checksumming,
  - Jumbo Frames,

- More peripherals:
Future work

- More networking features:
  - Pooling Mode,
  - Hardware Checksumming,
  - Jumbo Frames,
- More peripherals:
  - Pattern Matching Engine,
Future work

- More networking features:
  - Pooling Mode,
  - Hardware Checksumming,
  - Jumbo Frames,
- More peripherals:
  - Pattern Matching Engine,
  - Security Engine,
Future work

- More networking features:
  - Pooling Mode,
  - Hardware Checksumming,
  - Jumbo Frames,

- More peripherals:
  - Pattern Matching Engine,
  - Security Engine,
  - SATA Controller.
Acknowledgments
Acknowledgments

- Michał Dubiel (Semihalf),
Acknowledgments

- Michał Dubiel (Semihalf),
- Rafał Jaworowski (Semihalf, The FreeBSD Project),
Acknowledgments

- Michał Dubiel (Semihalf),
- Rafał Jaworowski (Semihalf, The FreeBSD Project),
- Phil Brownfield (Freescale)
Acknowledgments

► Michał Dubiel (Semihalf),
► Rafał Jaworowski (Semihalf, The FreeBSD Project),
► Phil Brownfield (Freescale)
► Zbigniew Bodek, Piotr Nowak, Tomasz Nowicki, Jan Sięka, Łukasz Wójcik (all Semihalf).
The End

Any questions?