KeyStone Training

C66x CorePac & Memory Subsystem
Agenda

• CorePac & Memory Overview
• Architecture Changes
• Multi-core Shared Memory Controller (MSMC)
• Memory Protection & Address Extension (MPAX)
• Extended Memory Controller (XMC) Prefetcher
• Memory Performance
• Summary
CorePac & Memory Overview

- CorePac & Memory Overview
- Architecture Changes
- Multi-core Shared Memory Controller (MSMC)
- Memory Protection & Address Extension (MPAX)
- Extended Memory Controller (XMC) Prefetcher
- Memory Performance
- Summary
CorePac & Memory Subsystem

- 1 to 8 C66x CorePac DSP Cores operating at up to 1.25 GHz
  - Fixed and Floating Point Operations
  - Code compatible with other C64x+ and C67x+ devices
- L1 Memory configured as cache
  - 32KB L1P per core
  - 32KB L1D per core
  - Error Detection for L1P
  - Memory Protection
- Dedicated and Shared L2 Memory
  - 512 KB to 1 MB Local L2 per core
  - 2 to 4 MB Multicore Shared Memory (MSM)
  - Multicore Shared Memory Controller (MSMC)
  - Error detection and correction for all L2 memory
  - MSM available to all cores and can be either program or data
- Boot ROM
Memory Expansion

Memory Subsystem
- 64-Bit DDR3 EMIF
- MSM SRAM
- MSMC

CorePac & Memory Subsystem

Memory Expansion
- Multicore Shared Memory Controller (MSMC)
  - Arbitrates CorePac and SoC master access to shared memory
  - Provides a direct connection to the DDR3 EMIF
  - Provides CorePac access to coprocessors and IO peripherals
  - Memory protection and address extension to 64 GB (36 bits)
  - Provides multi-stream pre-fetching capability
- DDR3 External Memory Interface (EMIF)
  - Support for 1x 16-bit, 1x 32-bit, and 1x 64-bit modes
  - Supports up to 1600 MHz
  - Supports power down of unused pins when using 16-bit or 32-bit width
  - Support for 8 GB memory address
  - Error detection and correction
- EMI-F-16 (Media Applications Only)
  - Three modes:
    - Synchronized SRAM
    - NAND flash
    - NOR flash
  - Can be used to connect asynchronous memory (e.g., NAND flash) up to 256 MB.

1 to 8 Cores @ up to 1.25 GHz

HyperLink

TeraNet

C66x™ CorePac
- L1 P-Cache
- L1 D-Cache
- L2 Cache

Application-Specific Coprocessors

Others
- P-C
- PCIe x2
- UART
- SPI
- Application-Specific I/O
- SRI x4

Ethernet Switch
- Security Accelerator
- Network Coprocessor

Queue Manager
- Packet DMA

Multicore Navigator

Texas Instruments
KeyStone Memory Topology

- **L1D** – 32KB Cache/SRAM
- **L1P** – 32KB Cache/SRAM
- **L2** - Cache/SRAM
  - 1024 KB for Wireless Apps
  - 512 KB for Media Apps
- **MSM** – Shared SRAM
  - 2048 KB for Wireless Apps
  - 4096 KB for Media Apps
- **L3** ROM – 128KB
- **DDR3** – Up to 8GB

L1D & L1P Cache Options – 0 KB, 4 KB, 8 KB, 16 KB, or 32KB
L2 Cache Options – 0 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB; 1024 KB for KeyStone Wireless applications only)
Overview: Features

- **Faster memory controllers and faster/wider busses** increase throughput to local and shared internal (MSMC RAM) /external memories (DDR3).
- **Shared internal memory** avoids duplication of critical common code and allows soft partitioning of internal memory between cores.
- **Prefetching** reduces the latency gap between local memory and shared internal/external memories.
- **Improved write merging and optimized burst sizes** reduce the stalls from/to external memory.
- **Memory protection** is extended to shared (internal/external) memories.
- **Address extension** provides a virtual address space for each core and extends the physical address space beyond 4 Gbytes using 36-bit addressing.
- Improved support for multi-core **cache coherency and memory consistency**.
Architecture Changes

- CorePac & Memory Overview
- Architecture Changes
  - Multi-core Shared Memory Controller (MSMC)
  - Memory Protection & Address Extension (MPAX)
  - Extended Memory Controller (XMC) Prefetcher
  - Memory Performance
- Summary
C64x+ to C66x Changes

Eliminates timing-critical "region decision" logic and L1DMPPA0-15

Eliminates L1P wait states and L1PMPPA0-15

CPU

L1D Memory Controller

Region 0

RAM only

Region 1

RAM/Cache

8 x 32

128

UMAP1

(RAM/SL2)

UMAP0

RAM/Cache

L1P Memory Controller

256

256

256

256

256

256

MARs

L2 Memory Controller

XMC

EMC

Prefetch

MPAX

IDMA0

IDMA1

CLK cvt

Width cvt

Config SCR

Convert from CLK/2 to CLK/1 for lower latency and higher peak throughput

Optimizes request patterns for better throughput

MDMA path grows from 128 bit @ CLK/3 to 256 bit @ CLK/2, separated from EMC and renamed to "XMC". XMC offers prefetch and MPAX (Memory Protection/Address Translation)

Converts from CLK/3 to CLK/1 for lower latency and higher peak throughput

Making MARs visible in L1D optimizes L1D write miss merging

Eliminates L2MPPA32-63 and second L2 mem. port

Eliminates L1P wait states and L1PMPPA0-15

Legend

CLK/1

CLK/2

CLK/3

Master input to MSMC controller

Slave input to system SCR

Convert from CLK/2 to CLK/1 for lower latency and higher peak throughput

Optimizes request patterns for better throughput

MDMA path grows from 128 bit @ CLK/3 to 256 bit @ CLK/2, separated from EMC and renamed to "XMC". XMC offers prefetch and MPAX (Memory Protection/Address Translation)
C64x+ to C66x Changes (2/3)

• New C66x Core
• Changes to L1P memory controller (PMC)
  – Region 0 removed (Not used in any C64x+ devices)
  – Eliminates L1PMPPA0-15 registers
• Changes to L1D memory controller (DMC)
  – Region 0 removed (Not used in any C64x+ devices)
  – Eliminates L1DMPPA0-15 registers
  – Enable write merging for external memory addresses
C64x+ to C66x Changes (3/3)

• Changes to L2 Memory Controller (UMC)
  – UMAP1 removed
    • Rem­oves as­so­ci­ated pipe­line logic
    • Elim­i­nates L2MPPA32-63 reg­i­sters
  – Pre-fecth sup­port added; Pre-fecth bit “PFX” added to MARs
  – Fa­s­ter clock: CPU­CLK in­stead of CPU­CLK/2.
  – Opt­i­mize re­quests for MSMC / DDR3 mem­ory
    • L2 line allo­ca­tions and evic­tions are split into sub­lines of 64 bytes
  – Im­proved pipe­line be­tween L1 and L2

• Changes to EMC
  – MDMA port moved to XMC
  – Ad­d­es abil­ity to map sys­tem PrivIDs to AIDs
  – Add PrivID remap­ping to han­dle sys­tems with large num­bers of mas­ters
New: Extended Memory Controller (XMC)

- Bring over existing EMC MDMA path
- Fat pipe to external (and internal) shared memory
  - Bus width: 256 instead of 128 bits
  - Clock rate: CPUCLK/2 instead of CPUCLK/3
- Memory Protection and Address Extension (MPAX) support
  - 16 segments of programmable size (powers of 2: 4KB to 4GB)
  - Each segment maps a 32-bit address to a 36-bit address.
  - Each segment controls access: supervisor/user, R/W/X, (non-)secure
  - Memory protection for shared internal MSMC memory and external DDR3 memory
- Multi-stream prefetch support
  - One forward L2/L1P miss stream prefetching up to 128 bytes of program
  - Eight forward/backward L2/L1D miss streams prefetching up to 128 bytes of data
  - Filter avoids picking up false L2/L1D miss streams
  - Prefetch enabled/disabled on 16MB ranges defined in MAR
  - Manual flush for coherence purposes
- NOTE: No IDMA path
System Level Enhancement: Fence

- Simple mechanism for programs to wait for CorePac’s requests to reach their endpoint
  - New instruction “MFENCE” that allows implementing fence operations
  - All read/write requests before “MFENCE” complete before any requests after “MFENCE”
  - Guarantees sequential consistency between groups of accesses
  - Convenient for exchanging data between multiple processors
  - This instruction will stall DSP until the completion of all the DSP-triggered memory transactions including the cache transactions.

- Provides ordering guarantees for
  - Writes arriving at a single endpoint via multiple paths
  - Multiprocessor algorithms that depend on ordering
  - Manual coherence operations
Fence Example: System Synchronization

- Two sets of stores to same endpoint (e.g. ASIC)
  - Writes could arrive in either order, due to different paths
  - Same problem even if you swap sets of stores
- Difficult to synchronize today, but not impossible
  - Depends on chip and which resources are being synchronized
- CorePac solution:
  - Insert “MFENCE” instruction
  - Should work regardless of chip or pair of resources
Multicore Shared Memory Controller (MSMC)

- CorePac & Memory Overview
- Architecture Changes
- **Multi-core Shared Memory Controller (MSMC)**
- Memory Protection & Address Extension (MPAX)
- Extended Memory Controller (XMC) Prefetcher
- Memory Performance
- Summary
Multicore Shared Memory (MSM) Subsystem

- **MSM Controller (MSMC)**
  - 256-bit wide direct-connect to EMIF-DDR (External Memory Interface)
  - 256-bit wide direct-connect to each DSP Core
  - Supports following cache options
    - Shared L2 SRAM mode (L1’s will cache, L2 will not cache requests to MSM SRAM)
    - Level 3 SRAM mode (L1 and L2 will both cache the MSM SRAM)

- **Multicore Shared Memory (MSM)**
  - Dynamically shared among all cores
  - Data and program
  - Error detection & correction
  - Memory protection

- **EMIF-DDR: DDR3 SDRAM Interface**
  - 1600 Mbps (800 MHz)
  - 1 x 64 bit
    - Additional topologies: 1x32 bit, 1x16 bit
    - 8-bit ECC per 64-bit data
  - 8 Gb of addressable memory

- **MSMC Prefetch Buffer**
  - Address extension from 32-bit to 36-bit address mapping
MSMC Feature Set

- Provides a common managed path to the shared internal memory (MSMC SRAM) for multiple CorePac cores and system masters
  - Optimized as L2 (cached in L1) memory
- Provides a common managed path to the shared external memory (DDR3) for multiple CorePac cores and system masters
  - DDR3 only, 1x16/32/64 data pins, up to 1.6 Gbps per data pin
  - Optimized as L3 (cached in L2) memory
- Provides a common path to chip level resources for multiple CorePac cores
- Provides memory protection and address extension / translation for accesses to the MSMC and DDR3 memory from non-CorePac system masters
- Features a dynamic fair-share bank arbitration for each transfer
- EDC support for MSMC memory, DDR3 EMIF already supports EDC for external memory
MSMC – Features Not Supported

• MSMC cache
• Cache coherency between L1/L2 caches in CorePac cores and MSMC memory
• Cache coherency between XMC prefetch buffers and MSMC memory
• CorePac to CorePac cache coherency via MSMC
• One slave interface per CorePac (256 bits @ CPUCLK/2)
  – Uses a 36 bit address extended inside a CorePac core

• Two slave interfaces (256 bits @ CPUCLK/2) for access from system masters
  – SMS interface for accesses to MSMC SRAM space
  – SES interface for accesses to DDR3 space
  – Both interfaces support memory protection and address extension

• One master interface (256-bits @ CPUCLK/2) for access to the DDR3 EMIF

• One master interface (256 bits @ CPUCLK/2) for access to system slaves
MPAX in MSMC

• System slave ports SES and SMS each feature an MPAX unit similar to that in the XMC for memory protection and address extension
• Divides the memory space into segments with associated permissions
  – Eight variable size segments for each privID, identified with a programmed base address and size
• Segment definitions can overlap
  – Definition in the highest numbered segment register has priority for address match i.e. segment register 5 prioritized over segment register 2
• Address extension/re-mapping support
  – MSMC SRAM address are clipped to keep the extended address within the SRAM range
  – DDR3 EMIF addresses are extended into the 36 bit address space
MPAX 
Segment 
Register 
Set 
Layout

MPAX register set for PrivID "0x0"

<table>
<thead>
<tr>
<th>SMS_MPAXH_0_0</th>
<th>SMS_MPAXL_0_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS_MPAXH_0_1</td>
<td>SMS_MPAXL_0_1</td>
</tr>
<tr>
<td>SES_MPAXH_0_0</td>
<td>SES_MPAXL_0_0</td>
</tr>
<tr>
<td>SES_MPAXH_0_1</td>
<td>SES_MPAXL_0_1</td>
</tr>
<tr>
<td>SES_MPAXH_0_2</td>
<td>SES_MPAXL_0_2</td>
</tr>
<tr>
<td>SES_MPAXH_0_3</td>
<td>SES_MPAXL_0_3</td>
</tr>
<tr>
<td>SES_MPAXH_0_4</td>
<td>SES_MPAXL_0_4</td>
</tr>
<tr>
<td>SES_MPAXH_0_5</td>
<td>SES_MPAXL_0_5</td>
</tr>
<tr>
<td>SES_MPAXH_0_6</td>
<td>SES_MPAXL_0_6</td>
</tr>
<tr>
<td>SES_MPAXH_0_7</td>
<td>SES_MPAXL_0_7</td>
</tr>
</tbody>
</table>

MPAX register set for PrivID "0xF"

<table>
<thead>
<tr>
<th>SMS_MPAXH_F_0</th>
<th>SMS_MPAXL_F_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SES_MPAXH_F_0</td>
<td>SES_MPAXL_F_0</td>
</tr>
<tr>
<td>SES_MPAXH_F_1</td>
<td>SES_MPAXL_F_1</td>
</tr>
<tr>
<td>SES_MPAXH_F_2</td>
<td>SES_MPAXL_F_2</td>
</tr>
<tr>
<td>SES_MPAXH_F_3</td>
<td>SES_MPAXL_F_3</td>
</tr>
<tr>
<td>SES_MPAXH_F_4</td>
<td>SES_MPAXL_F_4</td>
</tr>
<tr>
<td>SES_MPAXH_F_5</td>
<td>SES_MPAXL_F_5</td>
</tr>
<tr>
<td>SES_MPAXH_F_6</td>
<td>SES_MPAXL_F_6</td>
</tr>
<tr>
<td>SES_MPAXH_F_7</td>
<td>SES_MPAXL_F_7</td>
</tr>
</tbody>
</table>
MSMC Memory Banking

Bank 3

Sub-bank 1
2E0
1E0
E0

Sub-bank 0
2C0
1C0
C0

Bank 2

Sub-bank 1
2A0
1A0
A0

Sub-bank 0
280
180
80

Bank 1

Sub-bank 1
260
160
60

Sub-bank 0
240
140
40

Bank 0

Sub-bank 1
220
120
20

Sub-bank 0
200
100
00
MSMC Memory Banking Details

• LSB banked on 256-byte boundaries:
  – Four banks
  – Reduces conflicts between CorePac cores and system masters
  – Organized to work with XMC pre-fetch hardware access patterns
  – L2 line fill contained within one bank
• LSB sub-banked on 32-byte boundaries:
  – Two sub-banks
  – Sub-banks are 256-bit wide
  – Allows continuous back-to-back accesses to 1WS memory
  – Back-to-back 32-byte data phases of an L2 line fill alternate between the sub-banks of a bank
• Features a dynamic fair-share bank arbitration for each transfer
• Supports write transaction merging for transfers smaller than 256 bytes that are held up at bank arbitration
MSMC Bandwidth Management

- Avoid indefinite starvation for lower priority requests due to higher priority requests
  - MSMC features a bandwidth management scheme that limits starvation times.
- Starvation-Bound (SBND) register per requestor
  - SBNDC0-SBNDCn for the CorePac slaves
  - SBNDM for the SMS port
  - SBNDE for the SES port
  - Set a desired starvation bound in MSMC cycles for the requestor’s accesses.
- Each arbiter contains a starvation counter SCNT for each of the requestors being tracked.
- While a request is pending at the arbiter with a priority other than zero, the SCNT for the corresponding requestor is decremented till it reaches zero.
  - On the SCNT reaching zero, the priority of the request is elevated to zero (the highest priority level).
- Once the elevated priority is serviced, the SCNT counter is then re-loaded from the SBND register for the requestor.
  - Further accesses from the requestor will be based on the original priority.
Error Detect/Correct Support

- Hardware support for 2-bit error detection and 1-bit error correction for data stored in MSMC SRAM
- Parity generation for all ingress paths, correction/detection on egress paths
- Error correction support for both code and data
- Lower latency error detection mode available for CorePac data reads to tradeoff full error correction for performance
- Support for background data correction with a scrubbing engine
Memory Protection & Address Extension (MPAX)

- CorePac & Memory Overview
- Architecture Changes
- Multi-core Shared Memory Controller (MSMC)
- Memory Protection & Address Extension (MPAX)
- Extended Memory Controller (XMC) Prefetcher
- Memory Performance
- Summary
MPAX Units

- MPAX stands for Memory Protection and Address Extension
- There are N+2 MPAX units in a system with N CorePacs
  - N MPAX units for all requests from N CorePacs to internal shared memory, external shared memory or any system slave
  - One MPAX unit for all requests from any system master to internal shared memory
  - One MPAX unit for all requests from any system master to external shared memory
- Each MPAX unit operates on a number of segments of programmable size.
  - Each segment maps a 32-bit address to a 36-bit address.
  - Each segment controls access.
MPAX in the System
Number of Segments

• Each CorePac has 16 segments which control direct (load/store) requests to internal shared memory, external shared memory and any other system slave.

• Any master identified by a privilege ID has the following:
  – Eight segments for requests to internal shared memory
  – Eight segments for requests to external shared memory.

• Some masters work on behalf of other masters. They will inherit the privilege ID of their commanding master. As such, each CorePac also has the following:
  – Eight segments for indirect (DMA) requests to internal shared memory
  – Eight segments for indirect (DMA) requests to external shared memory

• The MPAX unit assigns a unique segment ID to each of its segments.
Segment Definition

• Each segment is defined by a base address and a size.
  – The segment size can be set to any power of 2 from 4K to 4GB.
  – The segment base address is constrained to power-of-2 boundary equal to size.
• One would expect that each request should find one matching segment, however
  – A request may find two or more matching segments, in which case segments with higher ID take priority over segments with lower ID. This allows the following:
    • Creating non-power of two segments
    • Creating three segments with just two segment definitions
    • ...
  – A request may find no matching segment, in which case an error is reported in Memory Protection Fault reporting registers (XMPFAR, XMPFSR).
Memory Protection

• The MPAX unit offers memory protection between CorePacs and system masters, not between tasks running on a single CorePac!
• The main goal is to protect the shared internal and external memories connected to the MSMC.
• All the other system slaves need to be protected with proper MPUs. The MPAX unit should be able to work with those other MPUs but is not intended to take over their role.
• Memory protection attributes are similar to CorePac’s MPA: supervisor/user, read/write/execute, secure/non-secure
Address Extension

• Each segment provides a replacement address.
  – The replacement address is constrained to power-of-2 boundary equal to the size.

• Expand from 4 GB to 64 GB address space.
  – Note that, even if the MPAX supports a 64 GB address space, Keystone devices supports external addressing only up to 8 GB.

• Map identical logical addresses to different physical addresses.
  – This may help the use of code that is shared between different CorePacs. Absolute references to private variables don't need to be redirected.

• Map different logical addresses to a single physical address.
  – This allows giving different semantics to the same memory. For instance, by having cacheable and non-cacheable access to a memory segment you can overcome the rough (16 MB) granularity of the MAR pages. Note that prefetching is also enabled/disabled per MAR page.
XMC Segment Registers: XMPAXH/XMPAXL[15-0]

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BADDR</td>
<td>Base Address</td>
<td>Upper bits of address range to match in C66x CorePac’s native 32-bit address space</td>
</tr>
<tr>
<td>SEGSZ</td>
<td>Segment Size</td>
<td>Segment size. Table below indicates encoding.</td>
</tr>
<tr>
<td>RADDR</td>
<td>Replacement Address</td>
<td>Bits that replace and extend the upper address bits matched by BADDR</td>
</tr>
<tr>
<td>PERM</td>
<td>Permissions</td>
<td>Access types allowed in this address range.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEGSZ</th>
<th>Meaning</th>
<th>SEGSZ</th>
<th>Meaning</th>
<th>SEGSZ</th>
<th>Meaning</th>
<th>SEGSZ</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000b</td>
<td>Seg. disabled</td>
<td>01000b</td>
<td>Rsvd (Disabled)</td>
<td>10000b</td>
<td>128KB</td>
<td>11000b</td>
<td>32MB</td>
</tr>
<tr>
<td>00001b</td>
<td>Rsvd (Disabled)</td>
<td>01001b</td>
<td>Rsvd (Disabled)</td>
<td>10001b</td>
<td>256KB</td>
<td>11001b</td>
<td>64MB</td>
</tr>
<tr>
<td>00010b</td>
<td>Rsvd (Disabled)</td>
<td>01010b</td>
<td>Rsvd (Disabled)</td>
<td>10010b</td>
<td>512KB</td>
<td>11010b</td>
<td>128MB</td>
</tr>
<tr>
<td>00011b</td>
<td>Rsvd (Disabled)</td>
<td>01011b</td>
<td>4KB</td>
<td>10011b</td>
<td>1MB</td>
<td>11011b</td>
<td>256MB</td>
</tr>
<tr>
<td>00100b</td>
<td>Rsvd (Disabled)</td>
<td>01100b</td>
<td>8KB</td>
<td>10100b</td>
<td>2MB</td>
<td>11100b</td>
<td>512MB</td>
</tr>
<tr>
<td>00101b</td>
<td>Rsvd (Disabled)</td>
<td>01101b</td>
<td>16KB</td>
<td>10101b</td>
<td>4MB</td>
<td>11101b</td>
<td>1GB</td>
</tr>
<tr>
<td>00110b</td>
<td>Rsvd (Disabled)</td>
<td>01110b</td>
<td>32KB</td>
<td>10110b</td>
<td>8MB</td>
<td>11110b</td>
<td>2GB</td>
</tr>
<tr>
<td>00111b</td>
<td>Rsvd (Disabled)</td>
<td>01111b</td>
<td>64KB</td>
<td>10111b</td>
<td>16MB</td>
<td>11111b</td>
<td>4GB</td>
</tr>
</tbody>
</table>
MSMC Segment Registers:
\( n=0..7 \) for each \( \text{PrivID}=0..15 \)

### Field | Name | Meaning
--- | --- | ---
BADDR | Base Address | The Segment Base Address field is used to match against the incoming address on the system slave port to identify the addressed segment. For SMS and SES, MPAXH[31:12] is compared against the requested address.
SEGSZ | Segment Size | Refer to the table "MPAX Segment Size Encoding" on page 1-8 for segment size encoding details.
RADDR | Replacement Address | Bits that replace and extend the upper address bits matched by BADDR
The BADDR and SEGSZ fields describe where each MPAX segment resides in the 32-bit address space for each CorePac.

The BADDR field indicates the start address of that segment in the logical address space for the corresponding CorePac.

A logical address resides within a given MPAX segment if its upper address bits match the corresponding bits in the BADDR field.
Address extension works by replacing the upper address bits of the logical address with corresponding bits from the replacement address (RADDR).

- RADDR[23:x] go to caddress_out[35:x+12]
- caddress_in[x+11:0] go to caddress_out[x+11:0]
- where caddress_in is the incoming address, and caddress_out is the output.
Address Extension Logic

• The table below indicates which bits of RADDR participate in address replacement according to the segment size.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00000b</td>
<td>Seg. disabled</td>
<td>01000b</td>
<td>Reserved</td>
<td>10000b</td>
<td>5</td>
<td>11000b</td>
<td>13</td>
</tr>
<tr>
<td>00001b</td>
<td>Reserved(1)</td>
<td>01001b</td>
<td>Reserved</td>
<td>10001b</td>
<td>6</td>
<td>11001b</td>
<td>14</td>
</tr>
<tr>
<td>00010b</td>
<td>Reserved</td>
<td>01010b</td>
<td>Reserved</td>
<td>10010b</td>
<td>7</td>
<td>11010b</td>
<td>15</td>
</tr>
<tr>
<td>00011b</td>
<td>Reserved</td>
<td>01011b</td>
<td>0</td>
<td>10011b</td>
<td>8</td>
<td>11011b</td>
<td>16</td>
</tr>
<tr>
<td>00100b</td>
<td>Reserved</td>
<td>01100b</td>
<td>1</td>
<td>10100b</td>
<td>9</td>
<td>11100b</td>
<td>17</td>
</tr>
<tr>
<td>00101b</td>
<td>Reserved</td>
<td>01101b</td>
<td>2</td>
<td>10101b</td>
<td>10</td>
<td>11101b</td>
<td>18</td>
</tr>
<tr>
<td>00110b</td>
<td>Reserved</td>
<td>01110b</td>
<td>3</td>
<td>10110b</td>
<td>11</td>
<td>11110b</td>
<td>19</td>
</tr>
<tr>
<td>00111b</td>
<td>Reserved</td>
<td>01111b</td>
<td>4</td>
<td>10111b</td>
<td>12</td>
<td>11111b</td>
<td>20</td>
</tr>
</tbody>
</table>

1. As noted previously, reserved encodings for the segment size disable the segment.
MPAX Reconfiguration

• MPAX reconfiguration is challenging because:
  • L1/L2 caches store logical addresses
  • L1/L2 caches store remote permissions
• As a result, the following procedure needs to be followed before the MPAX segment registers are modified:
  1. Write back and invalidate affected cache lines.
  2. Execute “MFENCE” instruction to ensure all write-backs have completed.
  3. Invalidate the prefetch buffer (using XPFCMD.INV) if prefetching is enabled for affected memory locations.
• Special care needs to be taken when fetching from the modified segment!
• As this process is costly, most programs will avoid changing MPAX segment registers often.
Memory Protection Fault Reporting Registers

Memory Protection Fault reporting registers enable programs to diagnose a memory protection fault after an exception occurs.

- **XMC:**
  - XMPFAR – XMC Memory Protection Fault Address Register
    - Access address causing the fault is captured
  - XMPFSR - XMC Memory Protection Fault Status Register
    - Records access type
  - XMPFCR – XMC Memory Protection Fault Clear Register
    - Allow clearing the fault information

- **MSMC:**
  - SMPFAR - Shared Memory Protection Fault Address Register
    - Access address causing the fault is captured
  - SMPFXR - Shared Memory Protection Fault Extended address Register
    - RADDR field from the MPAXL register is recorded
    - The master ID associated with the access is also recorded
  - SMPFR – Shared Memory Protection Fault Requestor Ids
    - PrivID of the faulting access is recorded, it contains one event bit per PrivID
    - It is set when an access from the PrivID faults
  - SMPFCR - Shared Memory Protection Fault Control Register
    - Allow clearing the fault information
    - There is only one fault recorded, the master would be notified with a fault interrupt and would receive a bus error for the access. In response, the master would need to clear the recorded fault before further faults can be recorded.
Multi-Core Virtual Memory

- Provides more external memory per core
- Isolates operating systems/applications running on different cores
- DOES NOT isolate processes running on the same core!
- Easily supports shared programs
MPAX Default Memory Map

- XMC configures MPAX segments 0 and 1 so that CorePac can access system memory.

- The power-up configuration is that segment 1 remaps 8000_0000 – FFFF_FFFF in Corepac address space to 8:0000_0000 – 8:7FFF_FFFF in the system address map.
  - This corresponds to the first 2GB of address space dedicated to EMIF by the MSMC controller.
Example shows three segments to map the MSMC RAM address space into CorePac address space as three distinct 2 MB ranges. By programming the MARs accordingly, the three segments could have different semantics.

Accesses to MSMC RAM via this alias do not use the “fast RAM” path and incur additional cycles of latency.
MPAX Overlayed Segments Example

- Segment 1 matches 8000_0000 through FFFF_FFFF, and segment 2 matches C000_7000 through C000_7FFF.

- Because segment 2 is higher priority than segment 1, its settings take priority, effectively carving a 4K hole in segment 1’s 2GB address space.

- Furthermore, it maps this 4K space to 0:5004_2000 - 0:5004_2FFF, which overlaps the mapping established by segment 2. This physical address range is now accessible by two logical address ranges.
MPAX Enables “Single Program Image”

- Example configuration for MSMC: Mix of shared program, shared data (cacheable and non-cacheable), per-Core private data

- Define private data by tying the MPAX to the DNUM, this way each core can have the private data separately in the physical memory, but still the logical address can be the same
XMC Prefetcher

- CorePac & Memory Overview
- Architecture Changes
- Multi-core Shared Memory Controller (MSMC)
- Memory Protection & Address Extension (MPAX)
- **Extended Memory Controller (XMC) Prefetcher**
- Memory Performance
- Summary
Prefetch What?

• Speculates which data (or instructions) the DSP will need
• Reads data (or instructions) from remote memory (MSMC RAM / DDR3) before the DSP needs them
• Stores data (or instructions) in prefetch buffer until the DSP needs them
Prefetching versus Caching

- Prefetch buffers are small - Cache memories are large.

- Prefetch controllers read data from memory before the DSP needs them - Cache controllers read data from memory when the DSP needs them.
  - Although, if DSP needs one part of cache line, the other part is prefetched.

- Prefetch controllers remove data from prefetch buffers when DSP needs them - Cache controllers keep data in cache memory when DSP needs them.
Why Prefetch?

• Good: Decrease read latency from memory
  – Latency to prefetch buffer is smaller than latency to remote memory
  – The DSP should see less stalls, thereby improves memory read performance to MSMC RAM and DDR3.

• Bad: Increase read traffic from memory
  – Every prefetch controller reads more data (or instructions) than needed
Prefetch Where?

• XMC has two Prefetch controllers
  – Data prefetch controller
  – Program prefetch controller

• The data prefetch controller reads data before the L1/L2 cache needs them

• The program prefetch controller reads instructions before the L1/L2 cache needs them
Multi-stream prefetching

• Data Prefetching
  – 8 forward/backward L2/L1D miss streams prefetching up to 128 bytes of data per stream
  – 12 filter slots available, it avoids picking up false L2/L1D miss streams

• Program Prefetching
  – 1 forward L2/L1P miss stream prefetching up to 128 bytes of program
Prefetch Streams: Good and Bad

• Not all streams will be happy:
  – Too steep: a stream making too big steps
  – Too slow: a slow stream among fast streams
  – Too many: a stream among many streams
  – Too bumpy: a stream with too many discontinuities
Controlling Prefetch Performance

• Setting / clearing MAR(N).PFX enables / disables prefetching for MAR page N
  – Each MAR register controls 16MB range
• Too high amplification of traffic / too low reduction of stalls?
  – move one or more streams from MAR page with prefetching enabled to MAR page with prefetching disabled
• Writing XMC:MDMAARBX.PRI sets the priority of the prefetch requests
  – never give prefetch requests higher priority than demand requests
Prefetch Coherency Issues

- Just like for caches, there may be coherency issues for prefetch buffers.
- Whenever you need to invalidate a cache line, check if you need to invalidate the prefetch buffer as well.
- XPFCMD.INV=1: invalidates the program stream and all data streams.
Memory Performance

- CorePac & Memory Overview
- Architecture Changes
- Multi-core Shared Memory Controller (MSMC)
- Memory Protection & Address Extension (MPAX)
- Extended Memory Controller (XMC) Prefetcher

- Memory Performance
- Summary
## Memory Read Performance

<table>
<thead>
<tr>
<th>Source</th>
<th>L1 cache</th>
<th>L2 cache</th>
<th>Prefetch</th>
<th>No victim</th>
<th>Victim</th>
<th>No victim</th>
<th>Victim</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Hit</td>
<td>NA</td>
<td>NA</td>
<td>0</td>
<td>NA</td>
<td>0</td>
<td>NA</td>
</tr>
<tr>
<td>Local L2 RAM</td>
<td>Miss</td>
<td>NA</td>
<td>NA</td>
<td>7</td>
<td>7</td>
<td>3.5</td>
<td>10</td>
</tr>
<tr>
<td>MSMC RAM (SL2)</td>
<td>Miss</td>
<td>NA</td>
<td>Hit</td>
<td>7.5</td>
<td>7.5</td>
<td>7.4</td>
<td>11</td>
</tr>
<tr>
<td>MSMC RAM (SL2)</td>
<td>Miss</td>
<td>NA</td>
<td>Miss</td>
<td>19.8</td>
<td>20.1</td>
<td>9.5</td>
<td>11.6</td>
</tr>
<tr>
<td>MSMC RAM (SL3)</td>
<td>Miss</td>
<td>Hit</td>
<td>NA</td>
<td>9</td>
<td>9</td>
<td>4.5</td>
<td>4.5</td>
</tr>
<tr>
<td>MSMC RAM (SL3)</td>
<td>Miss</td>
<td>Miss</td>
<td>Hit</td>
<td>10.6</td>
<td>15.6</td>
<td>9.7</td>
<td>129.6</td>
</tr>
<tr>
<td>MSMC RAM (SL3)</td>
<td>Miss</td>
<td>Miss</td>
<td>Miss</td>
<td>22</td>
<td>28.1</td>
<td>11</td>
<td>129.7</td>
</tr>
<tr>
<td>DDR RAM (SL2)</td>
<td>Miss</td>
<td>NA</td>
<td>Hit</td>
<td>9</td>
<td>9</td>
<td>23.2</td>
<td>59.8</td>
</tr>
<tr>
<td>DDR RAM (SL2)</td>
<td>Miss</td>
<td>NA</td>
<td>Miss</td>
<td>84</td>
<td>113.6</td>
<td>41.5</td>
<td>113</td>
</tr>
<tr>
<td>DDR RAM (SL3)</td>
<td>Miss</td>
<td>Hit</td>
<td>NA</td>
<td>9</td>
<td>9</td>
<td>4.5</td>
<td>4.5</td>
</tr>
<tr>
<td>DDR RAM (SL3)</td>
<td>Miss</td>
<td>Miss</td>
<td>Hit</td>
<td>12.3</td>
<td>59.8</td>
<td>30.7</td>
<td>287</td>
</tr>
<tr>
<td>DDR RAM (SL3)</td>
<td>Miss</td>
<td>Miss</td>
<td>Miss</td>
<td>89</td>
<td>123.8</td>
<td>43.2</td>
<td>183</td>
</tr>
</tbody>
</table>

**SL2** – Configured as Shared Level 2 Memory (L1 cache enabled, L2 cache disabled)  
**SL3** – Configured as Shared Level 3 Memory (Both L1 cache and L2 cache enabled)
Memory Read Performance - Summary

- Prefetching reduces the latency gap between local memory and shared (internal/external) memories.
  - Prefetching in XMC helps reducing stall cycles for read accesses to MSMC and EMIF DDR.
- Improved pipeline between L1D/L1P and L2 memory controller significantly reduces stall cycles for L1D/L1P cache misses.
- Performance hit when both L1 and L2 caches contain victims
  - Shared memory (MSMC or DDR) configured as Level 3 (SL3) have a potential “double victim” performance impact
- When victims are in the cache, burst reads are slower than single reads
  - Reads have to wait for victim writes to complete
- MSMC configured as Level 3 (SL3) is slower than Level 2 (SL2)
  - There is a “double victim” impact
- DDR configured as Level 3 (SL3) is slower than Level 2 (SL2) in case of L2 cache misses
  - There is a “double victim” impact
  - If DDR does not have large cacheable data, it can be configured as Level 2 (SL2).
## Memory Write Performance

<table>
<thead>
<tr>
<th>Source</th>
<th>L1 cache</th>
<th>L2 cache</th>
<th>Prefetch</th>
<th>No victim</th>
<th>Victim</th>
<th>No victim</th>
<th>Victim</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Hit</td>
<td>NA</td>
<td>NA</td>
<td>0</td>
<td>NA</td>
<td>0</td>
<td>NA</td>
</tr>
<tr>
<td>Local L2 RAM</td>
<td>Miss</td>
<td>NA</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MSMC RAM (SL2)</td>
<td>Miss</td>
<td>NA</td>
<td>Hit</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MSMC RAM (SL2)</td>
<td>Miss</td>
<td>NA</td>
<td>Miss</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MSMC RAM (SL3)</td>
<td>Miss</td>
<td>Hit</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>MSMC RAM (SL3)</td>
<td>Miss</td>
<td>Miss</td>
<td>Hit</td>
<td>0</td>
<td>0</td>
<td>6.7</td>
<td>14.6</td>
</tr>
<tr>
<td>MSMC RAM (SL3)</td>
<td>Miss</td>
<td>Miss</td>
<td>Miss</td>
<td>0</td>
<td>0</td>
<td>6.7</td>
<td>16.7</td>
</tr>
<tr>
<td>DDR RAM (SL2)</td>
<td>Miss</td>
<td>NA</td>
<td>Hit</td>
<td>0</td>
<td>0</td>
<td>4.7</td>
<td>4.7</td>
</tr>
<tr>
<td>DDR RAM (SL2)</td>
<td>Miss</td>
<td>NA</td>
<td>Miss</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>DDR RAM (SL3)</td>
<td>Miss</td>
<td>Hit</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>DDR RAM (SL3)</td>
<td>Miss</td>
<td>Miss</td>
<td>Hit</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>114.3</td>
</tr>
<tr>
<td>DDR RAM (SL3)</td>
<td>Miss</td>
<td>Miss</td>
<td>Miss</td>
<td>0</td>
<td>0</td>
<td>18.2</td>
<td>115.5</td>
</tr>
</tbody>
</table>

SL2 – Configured as Shared Level 2 Memory (L1 cache enabled, L2 cache disabled)
SL3 – Configured as Shared Level 3 Memory (Both L1 cache and L2 cache enabled)
Memory Write Performance - Summary

• Improved write merging and optimized burst sizes reduce the stalls to external memory.

• L1D memory controller merges writes to any (not only L2 RAM) address that is allowed to be cached (MAR.PC==1).
Summary

For more information, please refer to the following documentation:

• CorePac User Guide
• MSMC User Guide