# Simple MMU

## Overview

The SimpleMMU provides simple memory management capabilities for the Raptor64 CPU. Memory management by the SimpleMMU includes virtual to physical address mapping. The SimpleMMU divides a 128MB memory space up into 512 256kB pages and supports 32 tasks. Processor address bits 18 through 26 (the virtual address) are used as a nine bit index into a map table to find the physical address page. The MMU remaps the nine address bits into a 10 bit value used as address bits 18 to 27 when accessing a physical address. The lower eighteen bits of an address pass through the MMU unchanged. Also passing through the MMU unchanged are address bits 28 to 63. It is assumed that in the system where the Simple MMU would be relevant, that some or all of the high order bits of an address would be left unconnected. I/O accesses are not mapped by the SimpleMMU and I/O addresses pass through the MMU unchanged.

## Map Tables

The mapping table for memory management is stored directly in the SimpleMMU rather than being stored in main memory as is commonly done. The SimpleMMU directly supports up to 32 tasks. Each task has its own mapping table. The mapping table for only a single task is accessible at one time. Mapping table access is controlled by an access key. Eight MMU’s may be used in a system to allow up to 256 tasks.

## Access Key

Access to the mapping table is controlled by an access key. The access key contains the task number for the mapping table to be accessed. The mapping table for only a single task may be accessed at one time. In order to access a map table for another task, the access key must be updated with the desired task number. The upper three bits of the access key identify the MMU to be updated or read from. The lower five bits of the access key identify the map table.

## Operate Key

The operate key controls which map table (which task) is currently mapping addresses. The upper three bits of the operate key identify the MMU actively mapping addresses. The lower five bits select the map table within an MMU.

## Key Value Register

The key value register is used to identify the MMU and is how the MMU’s are differentiated. Operations on the MMU data are only possible if the key value register matches the high order three bits of the access key.

## Mapping Table / Register Set:

The register set for the MMU is based at I/O address of $DC4000. The mapping table appears as a set of 1024 consecutive I/O locations. All mmu’s share a common register set occupying the same I/O address range, with the exception of the key value register. Access to a particular mmu is controlled by the top three bits of the access key.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Reg | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |  |
| 00 | WP |  |  |  |  |  | PA27 | PA26 | PA25 | PA24 | PA23 | PA22 | PA21 | PA20 | PA19 | PA18 | | 512 map entries per task |
| 02 | WP |  |  |  |  |  | PA27 | PA26 | PA25 | PA24 | PA23 | PA22 | PA21 | PA20 | PA19 | PA18 | |
| 04 | WP |  |  |  |  |  | PA27 | PA26 | PA25 | PA24 | PA23 | PA22 | PA21 | PA20 | PA19 | PA18 | |
|  | … | | | | | | | | | | | | | | | | |
| 3FE | WP |  |  |  |  |  | PA27 | PA26 | PA25 | PA24 | PA23 | PA22 | PA21 | PA20 | PA19 | PA18 | |
| 400 |  |  |  |  |  |  | | |  |  |  |  |  | KV MMU0 | | | | Only one register per MMU |
| 402 |  |  |  |  |  |  | | |  |  |  |  |  | KV MMU1 | | | |
| 404 |  |  |  |  |  |  | | |  |  |  |  |  | KV MMU2 | | | |
| 406 |  |  |  |  |  |  | | |  |  |  |  |  | KV MMU3 | | | |
| 408 |  |  |  |  |  |  | | |  |  |  |  |  | KV MMU4 | | | |
| 40A |  |  |  |  |  |  | | |  |  |  |  |  | KV MMU5 | | | |
| 40C |  |  |  |  |  |  | | |  |  |  |  |  | KV MMU6 | | | |
| 40E |  |  |  |  |  |  | | |  |  |  |  |  | KV MMU7 | | | |
| 410 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | S | |  |
| 412 |  |  |  |  |  |  | | |  |  |  |  |  | Fuse | | | |  |
| 414 |  | | | | | | | | Access Key | | | | | | | | |  |
| 416 |  | | | | | | | | Operate Key | | | | | | | | |  |
| 418 |  | | | | | | | |  | | | | | | | | ME |  |

The top three bits of the access key must match the key value register in order to read/write the register set. Also, the ‘s’ bit must be set.

The lower five bits of the access key select the map for one of thirty-two tasks.

The operate key determines which task is the task actively mapping the address space.

The MMU divides memory up into 512 256k pages. Address bits 18 through 26 index into a map table to find the physical address page.

## Kernel Mode and the ‘s’ bit.

Transitioning into Kernel mode causes the ‘s’ bit to be set. This results in the MMU using task#0 to map addresses. The processor transitions into Kernel mode when a hardware interrupt or software exception occurs. In order to allow other tasks to map addresses, the countdown fuse must be set. When the countdown expires (it has to reach -1) the ‘s’ bit is cleared, and the task identified by the operate key controls memory mapping. The only way to clear the ‘s’ bit is by setting the countdown fuse. The ‘s’ bit is contained in a read-only register.

## Tasks

Task #0 is assumed to be the system task. Task #1 is assumed to be the DMA task. When the cpu transitions into kernel mode, task #0 is selected as the map controller. The ‘s’ bit is set which forces task #0 to map addresses. The task actively mapping addresses is controlled by the operate key when the ‘s’ bit is not set.

## Address Pass-through

Addresses pass through the MMU unaltered until the mapping enable bit is set. Until mapping is enabled, the physical address will match the virtual address. Additionally address bits 0 to 17 pass through the MMU unaltered. Address bits 28 to 63 pass through the MMU unaltered as well.