AMD

Mapping a Boot ROM on Alchemy™ Au1000™ Processor from AMD

Application Note

Revision: **1.2**
Issue Date: **January 2002**

Contacts

www.amd.com pcs.support@amd.com

Trademarks

1. Introduction

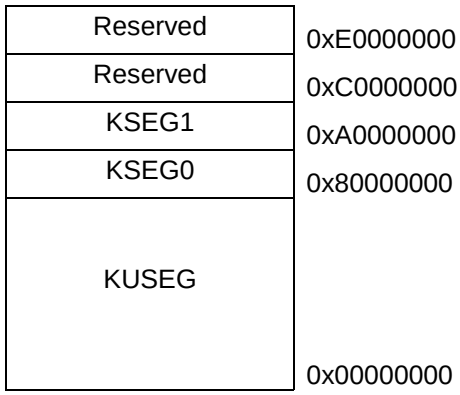
The Au1000™ processor contains integrated memory controllers for connecting RAM, ROM, and other peripherals to the Au1000 processor. This document describes how to map a boot ROM in a system based on the Au1000, and focuses on the use of the static memory controller for booting.

2. MIPS Architecture

The basic principles of where to map in a boot ROM are rooted in the MIPS architecture itself. Specifically, the MIPS architecture specifies that upon reset, a MIPS processor must fetch from address 0xBFC00000, the Reset exception vector. [1]

In the MIPS architecture, all addresses (instruction fetches, data loads and data stores) are virtual addresses. As a result, address translation is always performed on program instruction fetches and data accesses. The type of address translation depends upon the upper bits of the program address. The MIPS architecture defines the KUSEG, KSEG0 and KSEG1 regions according to these upper bits of the program's virtual address. The program's 32-bit memory space is thus divided:

Figure 1: MIPS 32-bit Memory Map



The KUSEG region extends from 0x00000000 to 0x7FFFFFFF, a 2GB space which uses translation look-a-side buffers, TLBs, to determine the corresponding physical address. The KUSEG region is accessible while the CPU is in either user mode or kernel mode.

The KSEG0 region extends from 0x80000000 to 0x9FFFFFFF, a 512MB space which has a direct correlation to a physical address. In addition, the KSEG0 region is inherently cacheable; meaning that both instruction and data caching is occuring for references to this area. The KSEG0 region is only accessible while the CPU is in kernel mode.

The KSEG1 region extends from 0xA0000000 to 0xBFFFFFFF, a 512MB space which also has a direct correlation to a physical address. However, the KSEG1 region is inherently non-cacheable; meaning that any instruction or data reference will bypass the cache and directly access physical memory. The KSEG1 region is only accessible while the CPU is in kernel mode.

For the KSEG0 and KSEG1 regions, the corresponding physical address is bits 28:0 of the virtual address with address bits 31:29 zero. That is, KSEG0 and KSEG1 map directly onto the first 512MB of physical memory. For example, KSEG0 address 0x80000000 and KSEG1 address 0xA0000000 both map directly onto physical address 0x00000000. The KSEG0 and KSEG1 regions provide two views of physical memory; one cacheable and one non-cacheable.

Thus, the MIPS architecture Reset exception vector address 0xBFC00000 lies in the KSEG1 region. This provides the MIPS processor a non-cacheable memory space in which to run while memory controllers, caches, TLBs and other system resources can be initialized properly before use.

The address translation mechanism of the MIPS architecture always presents a physical address to the memory controllers (and other address decode logic). Thus in the case of the Reset exception vector address 0xBFC00000, the physical address 0x1FC00000 is generated for the first instruction fetch from the boot ROM.

3. Mapping the Boot ROM on the Au1000

Au1000 processor-based systems typically use ROM and/or Flash memory connected to the static memory controller for booting the system. In the discussion that follows, the phrase “Flash memory” can be substituted “ROM” as appropriate. The examples below focus on booting the Au1000 from ROM on the static memory controller.

> *NOTE: The Au1000 processor can be configured to boot from either the static memory controller (via $\overline{RCE0}$) or the synchronous memory controller (via $\overline{SDCS0}$). The choice is determined by the signals ROMSEL and ROMSIZE at reset. Most of the following discussion is applicable when booting from synchronous memory devices connected to the synchronous memory controller. [2]*

3.1 Reset Conditions

On the Au1000 processor, the static memory controller chip-select $\overline{RCE0}$ is initialized at power-up to respond to physical address 0x1FC00000. More specifically, the mem_staddr0[E] is 0b1, mem_staddr0[CSBA] is 0b00011111110000, and mem_staddr0[CSMASK] is 0b11111111111111.

The mem_staddr0[E] bit enables the chip-select to respond if the physical address generated by the Au1 core is positively decoded by the CSBA/CSMASK criteria.

The mem_staddr0[CSBA] value establishes the base decode address of 0x1FC00000.

The mem_staddr0[CSMASK] value establishes a mask that permits a 256KB range off the base decode address.

Thus, out of reset, $\overline{RCE0}$ responds to 256KB of physical addresses in the range 0x1FC00000 thru 0x1FC3FFFF.

> *NOTE: If the processor performs a jump or branch to an address outside this initial 256K, the boot sequence will not succeed since $\overline{RCE0}$ will not respond. It is important to configure the static memory controller very early in the boot sequence to avoid this situation.*

3.2 Mapping the Boot ROM

Once the Au1 core begins to fetch instructions out of reset, it must configure the memory controllers to establish the memory map of the entire system. In particular, $\overline{RCE0}$ must be tailored for the appropriate boot ROM size installed in the system (see note above).

If the boot ROM is 256KB or less in size, then the mem_staddr0 register need not be changed. If the boot ROM is larger, then mem_staddr0 register must be changed accordingly, otherwise the entire boot ROM will not be visible to the Au1 core.

The MIPS architecture Reset exception vector is address 0xBFC00000 and is located in the last 4 Megabyte of the KSEG1 region. Thus the boot ROM effectively occupies the entire 4MB of space from 0x1FC00000 thru 0x1FFFFFFF.

3.2.1 Small Boot ROMs

For boot ROMs up to 4MB in size, the mem_staddr0[CSMASK] field must be updated to reflect the size of the boot ROM. The table below shows the value to use for CSMASK depending upon the boot ROM size.

Table 1: Small Boot ROM CSMASK Values

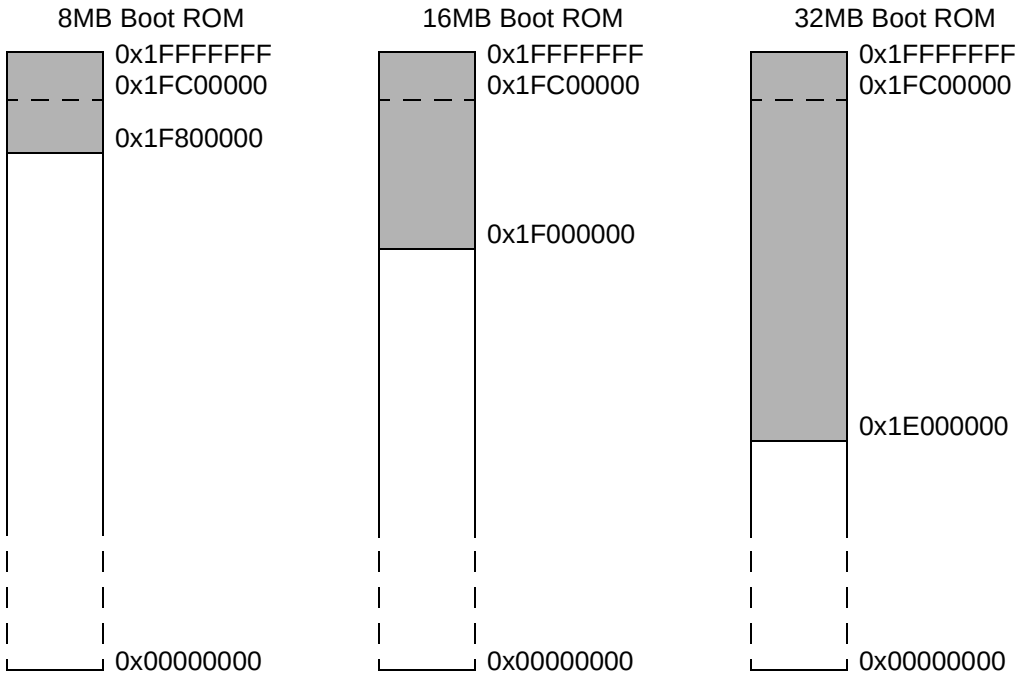
Boot ROM Size	CSMASK Encoding	CSMASK Value
<= 256KB	0b11111111111111	0x3FFF
512KB	0b11111111111110	0x3FFE
1024KB (1MB)	0b11111111111100	0x3FFC
2048KB (2MB)	0b11111111111000	0x3FF8
4096KB (4MB)	0b11111111110000	0x3FF0

For the small boot ROM, the Reset exception vector is at offset zero in the boot ROM and the entire boot ROM is mapped starting at physical address 0x1FC00000.

3.2.2 Large Boot ROMs

On the Au1000, however, it is possible to utilize boot ROMs much larger than 4MB. The technique for mapping boot ROMs larger than 4MB is this: rather than align the boot ROM to start at physical address 0x1FC00000, align the boot ROM to end at physical address 0x1FFFFFFF. This technique is depicted below:

Figure 2: Large Boot ROMs



To accomodate larger boot ROMs, mem_staddr0[CSBA] must change in addition to mem_staddr[CSMASK]. The table below shows the appropriate values to use for CSBA and CSMASK:

Table 2: Large Boot ROM CSBA and CSMASK Values

Boot ROM Size	CSBA Encoding	CSMASK Encoding
4MB	0b00011111110000	0b11111111110000
8MB	0b00011111100000	0b11111111100000
16MB	0b00011111000000	0b11111111000000
32MB	0b00011110000000	0b11111110000000
64MB	0b00011100000000	0b11111100000000
128MB	0b00011000000000	0b11111000000000

For the larger boot ROM, the Reset exception vector is no longer at offset zero in the boot ROM; rather the offset is at (boot ROM size - 4MB).

> *NOTE: When programming the boot code into the ROM using a part programmer, the size of the ROM changes the offset at which the Reset exception vector, and thus the boot code, must exist. For example, the exception vector for a 16MB boot ROM is located at offset 12MB into the boot ROM.*

The maximum size boot ROM size is 128MB since the next larger size overlaps with the internal peripherals.

3.2.3 mem_staddr0 Values

Combining the information from the small and large boot ROM tables, the table below gives the correct value to use for mem_staddr0 for the most common boot ROM sizes.:

Table 3: Boot ROM mem_staddr0 Values

Boot ROM Size	CSBA Encoding	CSMASK Encoding	Boot ROM Physical Base Address	mem_staddr0
<=256KB	0b00011111110000	0b11111111111111	0x1FC00000	0x11FC3FFF
512KB	0b00011111110000	0b11111111111110	0x1FC00000	0x11FC3FFE
1MB	0b00011111110000	0b11111111111100	0x1FC00000	0x11FC3FFC
2MB	0b00011111110000	0b11111111111000	0x1FC00000	0x11FC3FF8
4MB	0b00011111110000	0b11111111110000	0x1FC00000	0x11FC3FF0
8MB	0b00011111100000	0b11111111100000	0x1F800000	0x11F83FE0
16MB	0b00011111000000	0b11111111000000	0x1F000000	0x11F03FC0
32MB	0b00011110000000	0b11111110000000	0x1E000000	0x11E03F80
64MB	0b00011100000000	0b11111100000000	0x1C000000	0x11C03F00
128MB	0b00011000000000	0b11111000000000	0x18000000	0x11803E00

Of course, all registers in the static memory controller must be initialized upon boot to establish proper timings for peripherals, and to establish the memory map for the system.

4. References

[1] "MIPS32™ Architecture for Programmers", MIPS Technologies, Inc., 2001.

[2] "The Alchemy Au1000™ Internet Edge Processor Data Book", Alchemy Semiconductor, 2001.