1. CHANGING THE CACHEABILITY OF MEMORY FOR ZYNQ IN THE XILINX SDK

The purpose of this document is to summarize changing the cacheability of a memory region in run time. This allows different levels of cacheability on different memory regions while not requiring the user to alter the standalone BSP source files provided by Xilinx.

This document does not attempt to explain all the details of the page tables which are explained in the TRM.

2. STANDALONE BSP MMU INITIALIZATION

The standalone BSP sets up the page tables for the MMU such that DDR is set for inner (L1) and outer (L2) cacheable and shared. The shared bit (together with the SMP bit in the auxiliary control register) causes hardware coherency to be on, which causes the L1 caches of both CPUs and the L2 cache to be coherent.

This is done in source file translation_table.S of the ARM BSP which is located in <install>\SDK\2013.4\sw\lib\bsp\standalone_v3_12_a\src\cortexa9.

The following code snippet from the source file shows that 1024 32 bit words are being generated by the assembler. Each word is 0x15DE6 for the memory attributes and covers a 1 MB memory region. Each 32 bit generated is a level 1 page table entry for the MMU.

```
.rept   0x0400                  /* 0x00000000 - 0x3fffffff (DDR Cacheable) */
.word   SECT + 0x15de6          /* S=b1 TEX=b101 AP=b11, Domain=b1111, C=b0, B=b1 */
.set    SECT, SECT+0x100000
.endr
```

3. STANDALONE BSP FUNCTIONS

The standalone BSP provides a function, Xil_SetTlbAttributes, to allow the memory attributes to be modified for an MMU table entry.

The prototype of the function is provided in the source file xil_mmu.h of the ARM BSP which is located in <install>\SDK\2013.4\sw\lib\bsp\standalone_v3_12_a\src\cortexa9.

Multiple page table entries might need to be modified if the memory region to be altered is larger than 1 MB.

The attributes argument of the function call is a u32 and there are no macros defined to help set the specific cacheability bits.

4. PAGE TABLE ENTRIES FORMAT

The following diagram, taken from the TRM, shows the bit definitions for the level 1 page table entry. The section definition does not show the C and B bits and they are in the same location as shown for the supersection.

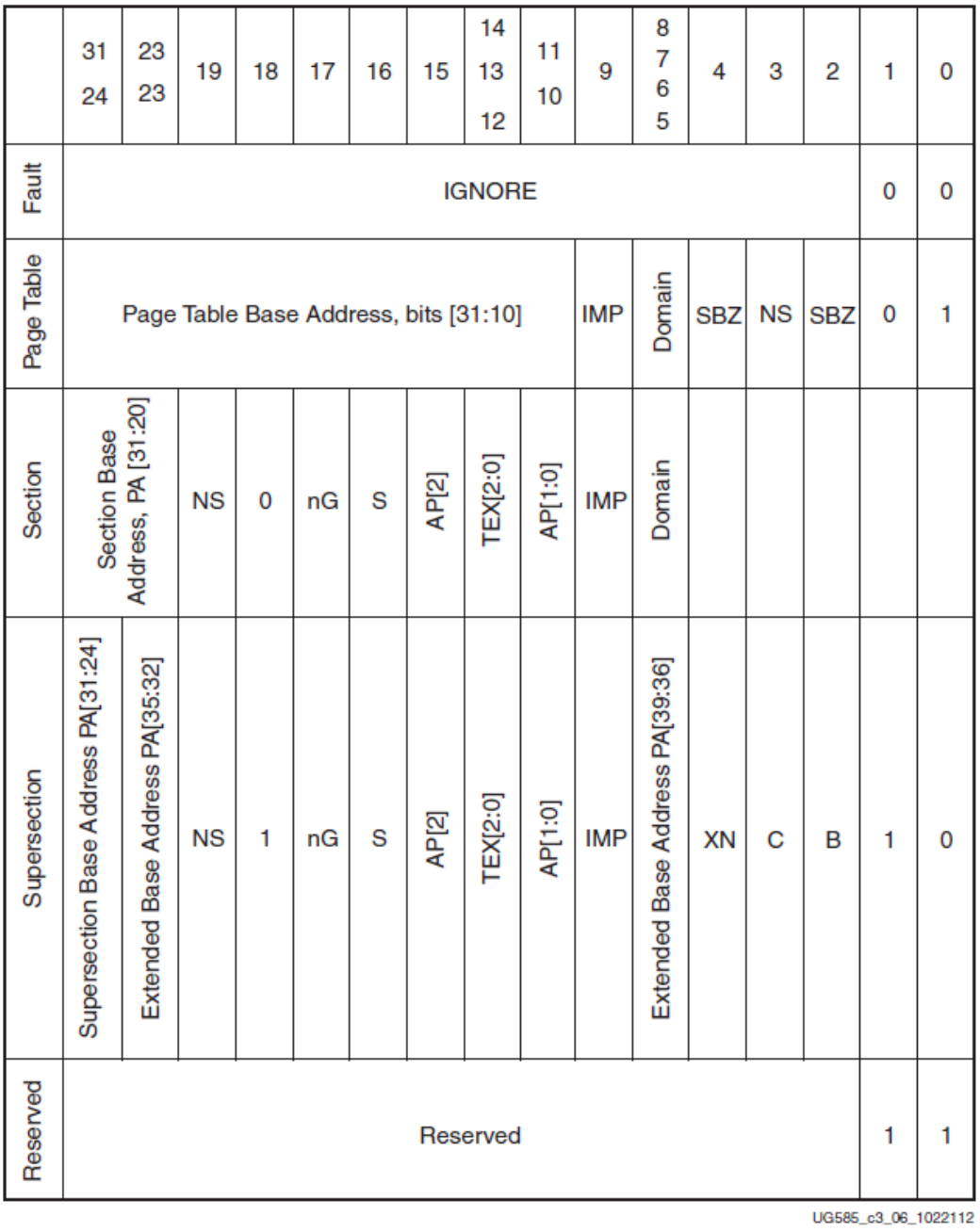

Figure 3-5: **L1 Page Table Entry Format**

5. MEMORY ATTRIBUTES

The following image was extracted from the TRM. The goal of putting these images into this document is to put all the information into one place so that it is easy to digest.

TEX, C, and B bits within the page table entry are used to set the memory attributes of a page and also the cache policies to be used. Memory attributes are discussed in 3.2.4 Memory Ordering, and for various cache policies please refer to the ARM Technical Reference Manual. Table 3-3 and Table 3-4 summarizes these memory attributes.

Table 3-3: **Memory Attributes Encodings**

TEX [2:0]	C	B	Description	Memory Type
0	0	0	Strongly-ordered	Strongly ordered
0	0	1	shareable device	Device
0	1	0	Outer and Inner write-through, no allocate on write	Normal
0	1	1	Outer and Inner write-back, no allocate on write	Normal
1	0	0	Outer and Inner non-cacheable	Normal
1	-	-	Reserved	-
10	1	0	Non-Shareable device	Device
10	-	-	Reserved	-
11	-	-	Reserved	-
1XX	Y	Y	Cached memory XX - Outer Policy YY - Inner Policy	Normal

Table 3-4: **Memory Attributes Encodings**

Encoding Bits		Cache Attribute
C	B	
0	0	Non-Cacheable
0	1	Write-Back, Write-Allocate
1	0	Write-Through, no Write-Allocate
1	1	Write-Back, no Write-Allocate

EXAMPLE OF CHANGING MEMORY ATTRIBUTES

The following code snippet defines the memory attributes so that they are a bit easier to understand and read. It has been tested minimally.

```
/* L1 Page Table Attribute definitions */

#define SHAREABLE              (1 << 16)

#define NON_CACHED             (0)
#define WB_WA_CACHE            (1)   /* write back, write allocate */
#define WT_CACHE               (2)   /* write thru */
#define WB_CACHE               (3)   /* write back */

#define INNER_SHIFT            (2)
#define OUTER_SHIFT            (12)
#define TEX_MSB                (1 << 14)
#define AP_FAULT               (0)         /* no access, a fault */
#define AP_FULL                (3 << 10)   /* full access */
#define DOMAIN(x)              (x << 5)

#define INNER_NON_CACHED       (NON_CACHED << INNER_SHIFT)
#define INNER_WB_WA            (WB_WA_CACHE << INNER_SHIFT)
#define INNER_WT               (WT_CACHE << INNER_SHIFT)
#define INNER_WB               (WB_CACHE << INNER_SHIFT)

#define OUTER_NON_CACHED       (TEX_MSB | (NON_CACHED << OUTER_SHIFT))
#define OUTER_WB_WA            (TEX_MSB | (WB_WA_CACHE << OUTER_SHIFT))
#define OUTER_WT               (TEX_MSB | (WT_CACHE << OUTER_SHIFT))
#define OUTER_WB               (TEX_MSB | (WB_CACHE << OUTER_SHIFT))

#define SECTION(attributes)               ((attributes) | 0x2)
#define SUPER_SECTION(attributes)         ((attributes) | 0x4002)

/* Set the OCM and DDR sections up to match what the standalone BSP does */

u32 OcmAttributes = SECTION(SHAREABLE | OUTER_NON_CACHED | AP_FULL | INNER_WB);
u32 DdrAttributes = SECTION(SHAREABLE | OUTER_WB_WA | AP_FULL | INNER_WB_WA
                                      | DOMAIN(0xF));

Xil_SetTlbAttributes(OCM_ADDRESS, OcmAttributes);
Xil_SetTlbAttributes(DDR_ADDRESS, DdrAttributes);
```