# FTMCTRL: 32-bit (P)ROM EDAC Checksum Programming

Application note 2018-04-17

Doc. No GRLIB-AN-0011

Issue 1.0

## CHANGE RECORD

| Issue | Date | Section / Page | Description |
|---|---|---|---|
| 1.0 | 2018-04-17 | All | First issue. |
| | | | |
| | | | |

## TABLE OF CONTENTS

# 1 INTRODUCTION

## 1.1 Scope of the document

This document describes programming of parallel EDAC checksum (also referred to as checkbits in this document) in systems that make use of the FTMCTRL memory controller. The focus is on programming checksums for non-volatile memories, specifically parallel NOR Flash, that require special address and data sequences to issue commands to the memory devices.

Parallel EDAC checksum is only used when EDAC protection is enabled with 32-bit data width.

## 1.2 Reference documents

[RD1] GRLIB IP Core User's Manual, Cobham Gaisler AB, https://www.gaisler.com/grip.pdf

[RD2] GRLIB-AN-0011-flash32 software package, available via https://www.gaisler.com/notes

# 2 ABBREVIATIONS

| | |
|---|---|
| BCH | Bose–Chaudhuri–Hocquenghem, class of cyclic error-correcting codes |
| EDAC | Error Correction And Detection |
| FTMCTRL | Fault-Tolerant Memory controller |
| MCFG | Memory Configuration Register, control register for memory controller |
| TCB | Test Check Bits, field in FTMCTRL MCFG2 register |

# 3 OVERVIEW

## 3.1 Overview

The FTMCTRL memory controller is commonly used in LEON3FT and LEON4FT processor devices and also in custom designs based on the GRLIB IP library [RD1].

The memory controller is a combined 8/16/32-bit memory controller that provides a bridge between external memory and the on-chip bus and is configured through memory-mapped registers referred to as the Memory Configuration (MCFG) registers. The memory controller can handle four types of devices: PROM, asynchronous static ram (SRAM), synchronous dynamic ram (SDRAM) and memory mapped I/O devices (IO). The PROM, SRAM and SDRAM areas can be EDAC-protected using a (39,7) BCH code. The BCH code provides single-error correction and double-error detection for each 32-bit memory word.

The PROM device type above typically means that parallel NOR Flash, MRAM or EEPROM is connected to the memory controller. A block diagram of how FTMCTRL can be connected to external devices and the on-chip system is shown in the figure below.

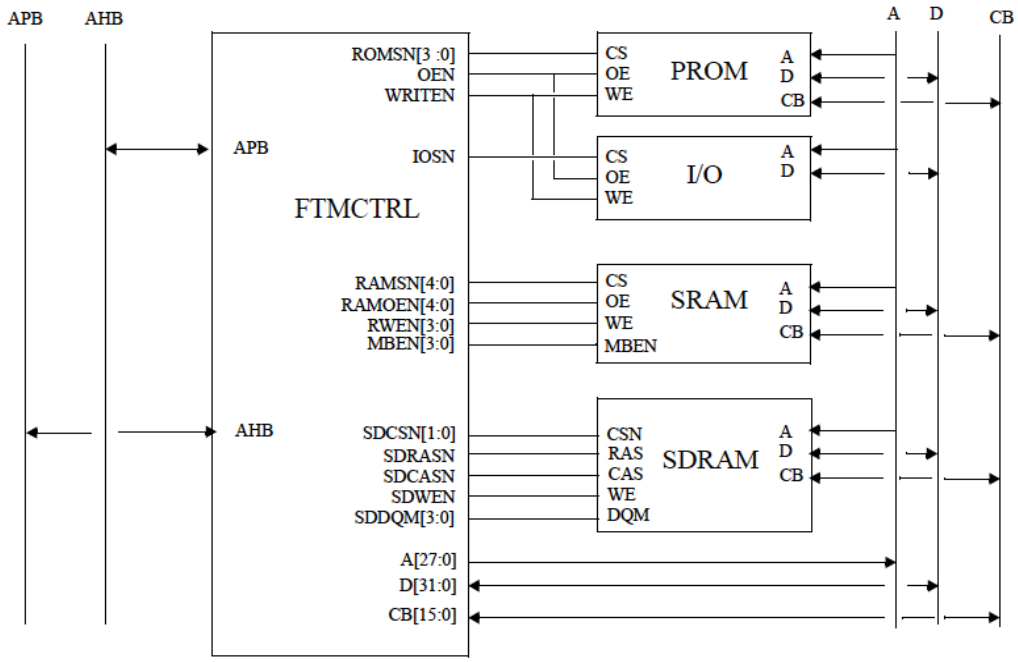


*Figure 1: FTMCTRL generic block diagram*

The types of devices supported and the signals available on external pins of a device depends on the specific device implementation.

## 3.2 FTMCTRL PROM EDAC

The FTMCTRL is provided with an BCH EDAC that can correct one error and detect two errors in a 32-bit word. For each word, a 7-bit checksum is generated. A correctable error will be handled transparently by the memory controller. If an un-correctable error (double-error) is detected, the current AHB cycle will end with an AMBA ERROR response. The EDAC is enabled for the PROM area by setting the corresponding EDAC enable bit in the MCFG3 register. When working in 32-bit mode, the checksum is present on the CB bus (see figure 1) and will be stored in a memory device present in parallel with the device(s) providing the 32-bit data bus.

For 8-bit mode, the EDAC checkbit bus (CB[7:0]) is not used but it is still possible to use EDAC protection. Data is always accessed as words (4 bytes at a time) and the corresponding checkbits are located at the address acquired by inverting the word address (bits 2 to 27) and using it as a byte address. Please refer to the relevant device user's manual or the FTMCTRL IP core documentation for further documentation on the 8-bit mode and EDAC.

When using a parallel device to hold the checkbits, the only way to set the data bus of that device to an arbitrary value is to use the write bypass and read bypass functionality provided by FTMCTRL. If the MCFG3.WB (Memory Configuration register 3, WB field - write bypass) bit is set, then the value in the MCFG3.TCB field will replace the normal checkbits during memory write cycles. If the RB (read bypass) is set, the memory checkbits of the loaded data will be stored in the TCB field during memory read cycles. This bypass functionality has some limitations:

- When read bypass is activated, then any memory read access will cause MCFG3.TCB to be updated.
- The read bypass functionality requires that EDAC is enabled.
- When write bypass is activated, then any memory write access will make use of the MCFG3.TCB field for the checksum value

This means that accesses to the memory controller must be limited in order for the read bypass and write bypass functionality to be reliable. The next section describes sources of memory accesses.

## 3.3 Sources of memory accesses

This document covers parallel checkbits for the PROM area. Since the same memory controller often provides provides access also to RAM memory used as the primary memory for a processor system, unwanted accessed may be caused by:

- Processor instruction fetches due to misses in the instruction cache
- Processor data fetches due to misses in the data cache
- Peripherals that perform direct-memory access (DMA)

### 3.4 Programming parallel checkbits

Programming of parallel checkbits is straightforward for memory types that accept write operations performed in the same way as a read operation, with the difference that a write signal is asserted.

Other devices, such as NOR Flash devices using the Common Flash Interface (CFI), require that both the address bus and the data bus are controlled when issuing commands to the memory devices and reading the responses to these commands. Controlling the data bus means that the write bypass functionality must be used in FTMCTRL and reading responses from a memory device means that the read bypass functionality needs to be enabled.

The read bypass and write bypass functionality of FTMCTRL can be used safely from a debugger such as GRMON by stopping the processor(s) and all on-chip peripherals capable of DMA in the system. If the bypass functionality shall be used from software running on the processor then it is possible to design a program, taking into account the cache structure and replacement policy of the processor implementation, that runs completely from cache. It is not possible to guarantee that the sequence will run from cache in an environment where radiation effects can case single-event upsets in the processor's cache or if interrupts are enabled which can lead to a changed flow of execution and changes in the cache state (and also to unintended write accesses from interrupt handling).

It should also be noted that a complicating, but not blocking, factor is that since read-bypass requires EDAC to be enabled, it is necessary to handle the corresponding AMBA ERROR, leading to a processor trap when reading CFI command responses via read-bypass.

Because of the limitations described above it is considered infeasible to perform CFI Flash programming with parallel checkbits from a processor that is executing from memory mapped to the same FTMCTRL, when using an operating system or when operating in an environment where L1 cache parity errors may be encountered.

### 3.5 Alternatives

Configurations with memory devices with 32-bit data and parallel checkbits may be wanted due to attainable memory size and memory access latency. In case the non-volatile memory devices need to be reprogrammed during operation then use of NOR Flash devices needs to be considered in combination with the limitations described in section 3.4. It can also be noted that if the non-volatile memory needs to be updated at random addresses then Flash devices usually only support erase operations on a page granularity. Alternatives to NOR Flash include MRAM devices and EEPROM devices.

A hybrid solution, usable unless the boot software needs to be updated, is to boot from FTMCTRL with EDAC enabled and make use of parallel checkbits. Once the system is up and running from RAM memory then the EDAC functionality for the PROM area can be disabled. EDAC for other parts of the PROM can then be implemented in software by creating a checksum for EDAC pages and storing it as part of the data that is memory-mapped. This way software will calculate and

validate checksums for the memory blocks that it reads and writes from non-volatile memory. The memory controller will not cause traps due to EDAC errors from the PROM after the EDAC is disabled.

# 4 CONTROLLING THE PARALLEL CHECKBIT BUS

The subsections below contain device specific observations and recommendations for CFI Flash programming.

## 4.1 UT699, UT699E and UT700

To safely read and control the parallel checkbit bus on a UT699 device from the LEON3FT processor, all accesses to the shared FTMCTRL memory controller must be controlled. This means that:

- All DMA units must be stopped
- Interrupts must be disabled
- Flash programming routines and their corresponding data must reside in L1 cache (cannot be guaranteed if L1 cache encounters parity-errors due to single-event upsets)

For the UT699 processor, L1 cache coherency through bus snooping cannot be used and this functionality will be disabled by software. For the UT699E and UT700 the cache snooping functionality can optionally be enabled by software. If bus snooping is enabled then snooping will invalidate cache lines due to DMA traffic and this could have effects for software implementations that rely on data being present in cache for PROM programming.

## 4.2 GR712RC

For software running out of external memory, the same limitations apply as described for the UT699E and UT700 in section 4.1.

Many of the limitations come from the need to execute software from the same memory controller as the one that provides access to the external non-volatile memory. The GR712RC also has an on-chip RAM. If this RAM is utilized to hold the programming application then it is sufficient if the following rules are met:

- All DMA units using external memory must be stopped
- The full program, including trap table, must reside in the on-chip RAM
- An adapted trap handler for handling AMBA ERROR responses caused by reading memory device command responses with read-bypass must be installed.

A software example for programming NOR Flashes with parallel checkbits is available [RD2].

### 4.3 GR740

The FTMCTRL in GR740 supports 8- and 16-bit interfaces. EDAC check bits are programmed in the memory-mapped area and the special precautions described in this document do not need to be considered for the GR740.

### 4.4 LEON3FT-RTAX

The same restrictions as the ones listed in section 4.1 apply.

### 4.5 Other designs with FTMCTRL

The same restrictions as the ones listed in section 4.1 apply for devices that has one FTMCTRL that provides access to both RAM and non-volatile memory. For devices that have other RAM or ROM that software can use, the restrictions described in 4.2 apply.

## 5 GRMON

GRMON versions 1.x.y and 2.x.y do not support programming parallel check bits. Support will be added for GRMON3 and this document will be updated with version information once the feature is available in GRMON3.