Using 8-bit PROM and EDAC with the UT699

Table 1: Cross Reference of Applicable Products

Product Name:	Manufacturer Part Number	SMD #	Device Type	Internal PIC* Number:
UT699 32-bit Fault-Tolerant SPARC V8/LEON 3FT Processor	UT699	5962-08228	01, 02	WG07

* PIC = Product Identification Code

1 Overview

System designers can use an 8-bit PROM with the UT699 either for simplicity of a design or to minimize the amount of required PCB space. When Error Detection and Correction (EDAC) is utilized in such a system, the PROM contains both the program memory and the EDAC checksum bytes in the same device. This application note explains how the EDAC check-sums are stored and how the PROM should be connected to the UT699.

2 Hardware Configuration

Designers should connect an 8-bit PROM to the UT699 as shown in Figure 1. The same hardware configuration applies whether or not EDAC is used except that only $\overline{\text{ROMS[0]}}$ is used when EDAC is enabled. Data bus D[31:24] of the UT699 is connected to D[7:0] of the PROM. The address bits of the UT699 are connected to the corresponding address bits of the PROM beginning with A[0]. The EDAC check bit bus CB[7:0] is not used in the case of 8-bit PROM. The memory controller of the UT699 drives the checksums onto D[31:24].

One consideration must be made in regard to the size of the program memory. When using EDAC, the lower 80% of the PROM contrains the program memory while the upper 20% is used to store the EDAC checksums. Therefore, only 80% of the PROM can be used to store the program or data memory.

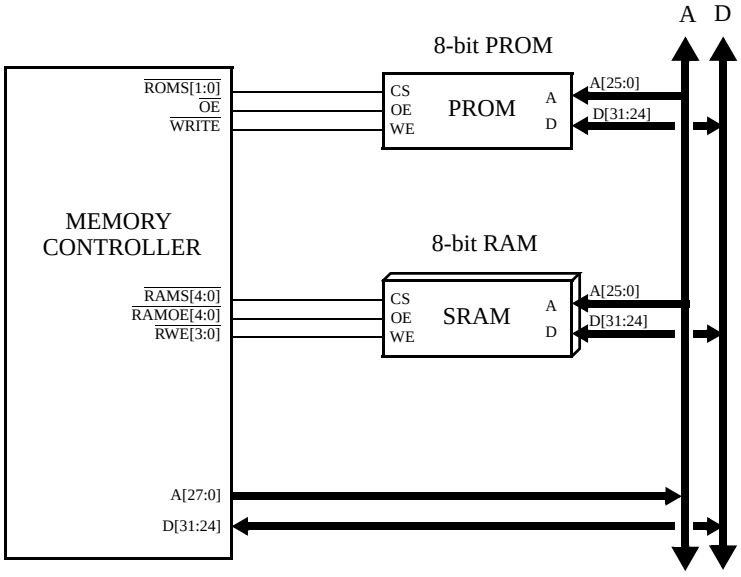

Figure 1. PROM Interface in 8-bit Mode

3 PROM Data Storage

Program memory in PROM begins at address 00000000_{16}. Each instruction is 32 bits wide, occupying four bytes of memory, with one additional byte required for the EDAC checksum. Therefore, an instruction fetch requires five CPU clock cycles in the case of 8-bit PROM with EDAC enabled. The UT699 uses a big-endian architecture, meaning that the address of the instruction is also the address of the most-significant byte (MSB). The EDAC check bytes are stored top-down starting at the top of the address of the PROM. Figure 2 shows an example of the first two instructions in PROM and their corresponding EDAC check byes.

The address of an EDAC check bytes for an instruction is calculated as follows:

- Bit-shift right twice the address of the MSB of the instruction
- Take the one's complement of the result

ADDR	DATA[31:24]
$FF...FFFF_{16}$	Checksum 0
$FF...FFFE_{16}$	Checksum 1
...	...
...	...
...	...
$00...0007_{16}$	Instruction 1 [7:0]
$00...0006_{16}$	Instruction 1 [15:8]
$00...0005_{16}$	Instruction 1 [23:16]
$00...0004_{16}$	Instruction 1 [31:24]
$00...0003_{16}$	Instruction 0 [7:0]
$00...0002_{16}$	Instruction 0 [15:8]
$00...0001_{16}$	Instruction 0 [23:16]
$00...0000_{16}$	Instruction 0 [31:24]

Figure 2. PROM Instruction Data and EDAC Check Byte Storage

As an example, consider the first two instruction words shown in Figure 2. The PROM selected is an 8-bit device with 1MB of storage capacity requiring 20 address lines. The addresses of the first two instructions are 00000_{16} and 00004_{16}. The address of the first EDAC checksum is:

- $00000_{16} >> 2 = 00000_{16}$
- $\sim 00000_{16} = FFFFF_{16}$

The address of the second EDAC checksum is:

- $00004_{16} >> 2 = 00001_{16}$
- $\sim 00001_{16} = FFFFE_{16}$

When the UT699 takes the one's complement of an address, all address bits are affected regardless of the PROM size. For example, in the case of the second instruction the UT699 drives an address pattern of $FFFFFFF_{16}$ for the check byte. Although all 28 address bits are driven, only the 20 least-significant bits are tied to the PROM. The upper eight bits are driven but not used.

4 PROM File Generation

Software developers can use the *mkprom2* utility to generate an 8-bit PROM file containing the EDAC checksums from code compiled with the BCC compiler. First, compile the code and generate an object file:

```
sparc-elf-gcc -c hello.c
```

The compiler will generate the file *hello.o*. Next, run the *mkprom2* utility with the following options:

```
mkprom2 -romwidth 8 -romcs 1 -bch8 -romsize 1024 hello.o
```

The *-bch8* option instructs *mkprom2* to generate the file *prom.out.bch8*, which will place the checksums in the top 20% of the PROM. Options *-romwidth 8* and *-romcs 1* are required, and indicate a PROM width of eigth bits and that only one PROM chip select is used. The *-romsize* option indicates the size of the PROM in kB. The *mkprom2* utility is available at www.gaisler.com.

5 Conclusion

The UT699 can easily interface to an 8-bit PROM and take full advantage of its EDAC capability. This configuration offers a simple solution where board space is a limiting factor. Software development tools are readily available to facilitate a design utilizing an 8-bit PROM with EDAC.