Integrated Device Technology, Inc.

ERROR DETECTION AND CORRECTION WITH IDT49C466

APPLICATION NOTE AN-94

By Anupama Hegde

INTRODUCTION

It is widely accepted that system failures and down time claim a heavy toll in terms of cost and performance. When a system crashes, unrecoverable data may be lost. Even in the best case, the user suffers a great deal of inconvenience. Erroneous data or loss of data integrity is a major cause of system failure. Hence system designers are constantly trying to minimize the occurrence of errors in their systems.

Data errors commonly occur in storage devices such as RAM, disks and magnetic tapes. Hence error handling schemes are commonly aimed at protecting the data in these devices. Dynamic memory is, in particular, highly susceptible to errors. As we move towards systems with larger memories, data protection schemes become increasingly relevant because the error rate is found to increase with the memory size. Till recently parity has been the traditional method of data protection. Parity, however, has several drawbacks: it masks out even bit errors, it cannot locate the bit in error and beyond a certain buswidth it can become inefficient. These factors have lead to the development of a new breed of devices for data protection. Error Detection and Correction circuits (EDCs or EDACs) are today's response to the growing need for greater data reliability.They afford a greater level of protection than that offered by parity, by using a higher order error correction code (Modified Hamming code).

Implementing error detection & correction in a system involves not just the hardware to perform the error checking but also extra memory for storage of "checkbits", which are the basis of error detection and correction. An EDC may add some overhead to a system in terms of speed and cost. This is, however, compensated by a significant improvement in system performance. The following Figures underline this point:

Given a 4M x 64 DRAM memory with an FIT(Failure in time) specification of 252^1:

MTBF (mean time between failures) without EDC = 1.76 years
MTBF with single-bit error correcting 64-bit EDC = 3935.4 years

The Figures given here are merely a guideline for comparison of the two cases and will depend entirely on the DRAM used and the memory configuration.

Thus EDC can be looked upon as a sophisticated parity system with the capacity to correct. Two primary functions are performed by an EDC unit within the framework of the memory read and write cycles - during a memory read, errors are detected and/or corrected and during a memory write, checkbits are generated.

ERROR DETECTION AND CORRECTION WITH THE 49C466

Choosing the Right EDC

EDC considerably upgrades reliability but, as mentioned earlier, there is a certain amount of overhead associated with implementing EDC. This is because of the checkbit memory required and the nanoseconds it adds to the main memory cycle times. The checkbit overhead can be minimized by increasing the size of the data word used to generate the checkbits. The table below illustrates this point and shows how parity compares with EDC for various data word sizes.

Assuming a distance-of 4 Hamming code for the EDC and byte parity:

As an example, consider a 32-bit EDC and a 64-bit EDC. We see from the table above, the former requires 7 checkbits

DATA WORD SIZE	PARITY BITS REQUIRED	EDC CHECKBITS REQUIRED
16	2	6
32	4	7
64	8	8
128	16	9

2596 tbl 01

for each 32-bit data word. Hence every 64 bits of data requires 7+7=14 checkbits in memory. A 64-bit EDC on the other hand requires only 8 checkbits to provide single bit error correction and dual bit error detection. Thus, as far as checkbit efficiency goes, a 64-bit EDC is better than a 32-bit EDC. One other advantage of a wider EDC is that less checkbit memory implies a lower error rate in checkbit memory. Going after even wider EDCs seems tempting in this light, but a compromise has to be made because of two factors.

Prevalent system bus widths impose a practical limitation on how wide a bus can get. It is fairly common nowadays, however, to see a 64-bit memory bus and hence a 64-bit EDC is often easily accomodated. An EDC that is wider than the memory bus needs additional control and logic circuitry to integrate the EDC with the memory system. The greater this difference in (EDC and memory) bus widths, the more complex is this interface. Also, the error handling capacity effectively decreases as the EDC bus gets wider because the same level of protection (1 bit correction,2 bit detection) is provided for a wider word. Two 32-bit EDCs provide this for each 32-bit word, which allows correction of some 2-bit errors and detection of some 4-bit errors for each 64-bit word. Thus, constraints for a given system determine the optimum trade-off.

NOTE: See appendix for further details on calculation of MTBFs.

49C466 Operation and Features

By the nature of its function, the EDC is concerned with data transfers between the processor and memory. The IDT 49C466 has a flowthrough architecture which permits transparent data flow through the EDC. The 49C466 also increases checkbit efficiency by providing two 64-bit wide bidirectional data buses.

The error detection & correction operation in the 49C466 is similar to previous generation IDT devices. A modified Hamming code is used to generate the checkbits and the syndrome decoding is identical to cascaded mode operation in the 32-bit EDC, the 49C465. The modified Hamming code used allows for flagging of all single , dual and three bit errors in the 64-bit data word. Some three bit errors alias as single bit errors, however, and may hence be wrongly decoded by the syndrome decoding logic (MERR* may not be asserted). The code allows for correction of all single bit errors in the data word.

The 49C466 has five modes of operation. In the *normal* mode, two kinds of operations are performed. During a "memory write", checkbits are generated based on the data that is written and during a "memory read" single-bit errors in the data are corrected. In the *detect-only* mode, the "memory write" operation remains the same but during a "memory read" any errors detected are flagged by $\overline{\text{ERR}}$ and $\overline{\text{MERR}}$ pins. The data is passed though unaltered. The other three modes are useful for testing & diagnostic purposes. The EDC mode of operation is set by the user by loading the mode register through the system data bus.

Read and write paths are independent of each other in the 49C466. In fact, the device has alternate paths for each of these (read and write cycles). One path includes the 8/16-deep data buffer while the other provides a latch in place of this buffer. WBSEL(Write Buffer Select) and RBSEL(Read Buffer Select) pins select the output of either the buffer or the latch (for example - RBSEL=high selects output from the Read Buffer rather than the MD_OUT latch, WBSEL = 0 selects the output of the SD_IN latch rather than the Write Buffer).

Memory Write

During a memory write, the EDC generates checkbits corresponding to the data being written to memory (data flow from SD bus to MD bus). These are output onto the CBSYN bus and written to checkbit memory. This is a necessary part of EDC operation. Unless all checkbits corresponding to the data in memory have been generated and stored in checkbit memory, no error checking is possible.

To prevent contention on the SD bus, the SD output buffer must be disabled, during a memory write. The MD and checkbit output buffers must be enabled to pass data and checkbits to memory.

There are several ways that the "write path" can be configured but the maximum "write time" with input and output latches transparent is 15ns.

Figure 2 illustrates the data path during a write operation. The alternate write path, where the data is buffered, is shown in Figure 3 .

Memory Read

A memory read involves checking data from memory for errors. Given below is a description of what actually takes place during a "read cycle".

Data and checkbits from main memory are fed to the EDC. Checkbits corresponding to the read data are generated within the EDC. The two sets of checkbits are internally compared to produce the syndrome word. The EDC then decodes the syndrome word to check for errors.

The syndrome is also clocked into the syndrome register on the rising edge of SYNCLK. The syndrome register contents can be output on the CBSYN bus. The CBSEL pin controls the output of the CBSYN bus. While CBSEL is low, checkbits generated during a write are output. When CBSEL is high, syndrome generated during a read are output. Figure 4 shows the data path during a normal read operation.

A buffered read operation is illustrated in Figure 5.

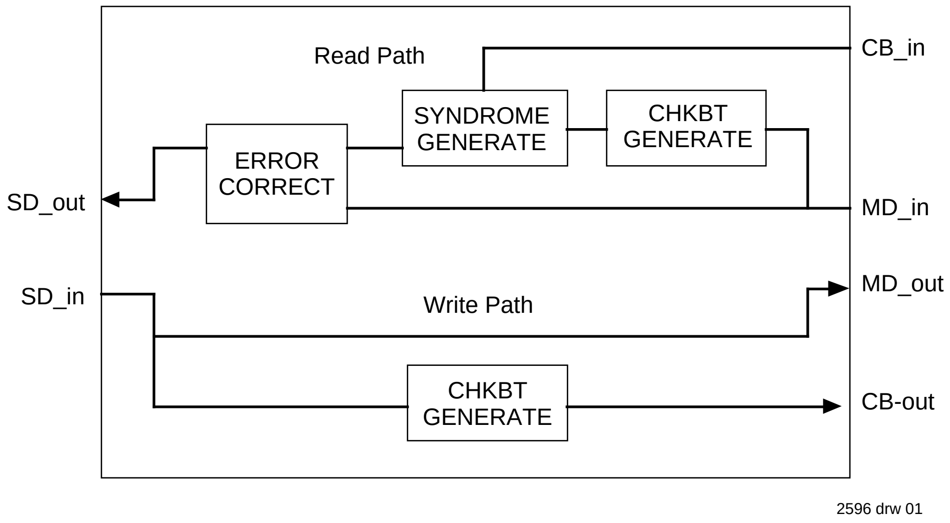


Figure 1. Basic Memory Read and Write paths Through the 49C66

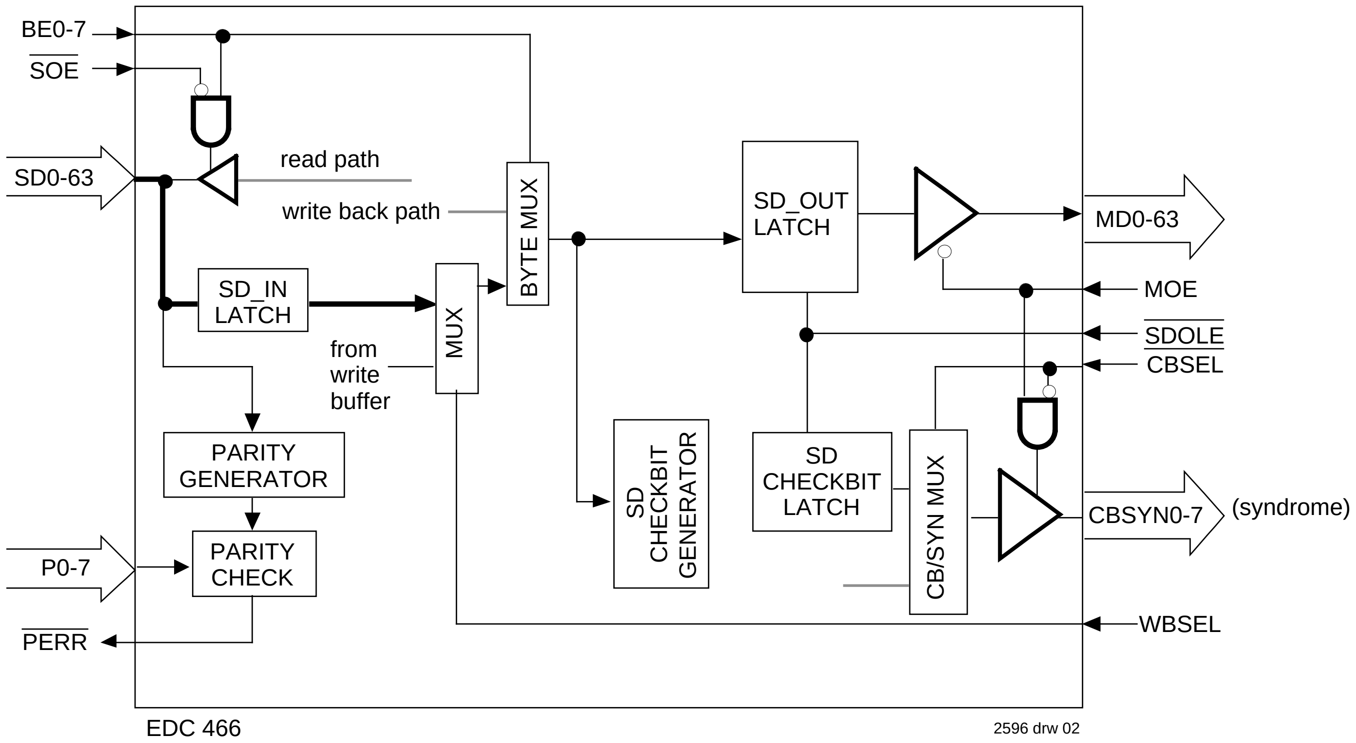


Figure 2. Memory Write without Write Buffer

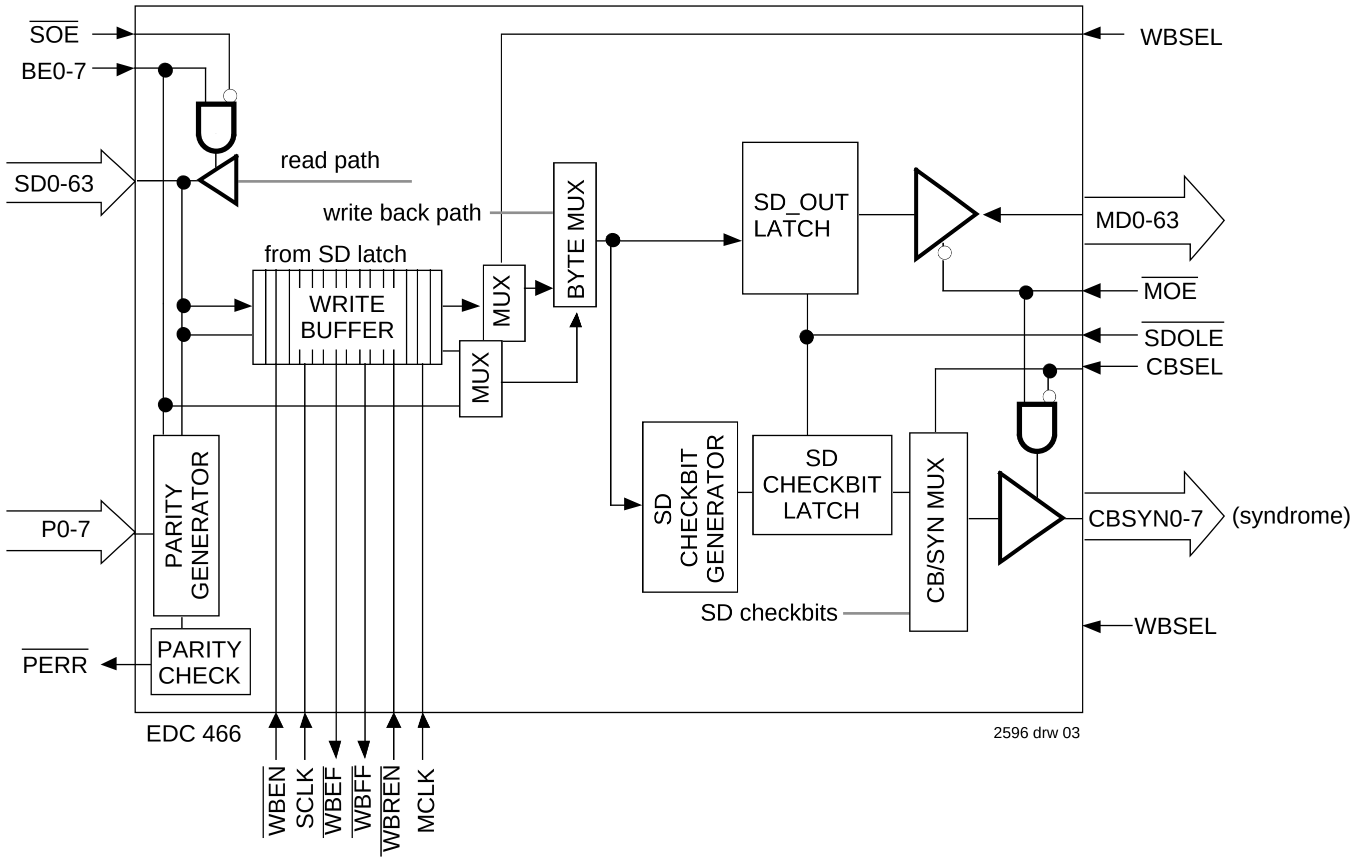


Figure 3. Memory Write with Write Buffer

Beside its primary task of error checking, the 49C466 integrates certain other useful functions on chip. It supports parity checking and generation, partial word writes and diagnostics. It provides the user with flexibility by providing 8/16 deep Read and Write buffers and the facility to latch all data flowing in and out of the part.

The parity generation and checking capability in the 49C466 is very useful in checking the integrity of the data being written to memory. Parity checking is done on data from the system side. The parity bits are input on the EDC P0-7 pins. Parity is then generated for the input system data and a comparison of this and the input parity bits is carried out internally. The result of this comparison is reflected in the $\overline{\text{PERR}}$ pin output (a discrepancy asserts $\overline{\text{PERR}}$). Parity bits are also generated for data read from memory and output on the P0-7 pins once again. The parity type (odd or even) is selected using the PSEL pin. The $\overline{\text{PERR}}$ signal does not affect any of the other circuitry in the device and hence the user may safely ignore the parity feature if his system does not support parity.

PARTIAL WORD WRITES

The 49C466 supports "partial word writes" or "byte merging". These refer to write operations involving words shorter than 64 bits such as a byte write. Partial word writes are handled in the 49C466 by the byte multiplexer. The 49C466 has eight BE (byte enable) control pins. These control the byte multiplexer and enable the system data output buffer. When a BE input is high, it selects a byte from the MD "write back path" shown in Figure 3 and Figure 4 rather than the normal EDC write path. Each BE input is AND-ed with the $\overline{\text{SOE}}$ signal and the result determines which byte is output on the SD bus. A BE pin that is low disables the SD output buffer for the particular byte referenced and selects a byte from the write buffer or SD_in latch rather than one from the MD write back path.

A partial word write or byte merge is analogous to a read-modify-write cycle. The checkbit word generated by a partial data word would be incomplete and hence incorrect. Hence the need to differentiate the "partial word write" case from a normal write operation. The following steps must be followed to ensure correct generation of checkbits for the complete 64 bit word :

1. Read the contents of the location being written to.
2. Merge the partial word with these contents forming a composite 64 bit data word.
3. Generate the 8 checkbits for this composite word.

On account of the dual bus feature of the 49C466, data may be written to the EDC (from processor/cache) while data is read from main memory to the EDC. This feature is useful during partial word writes (writing less than eight bytes) and can buy some time for the designer for whom each nanosecond counts. The time required to turn the external buses around must, however, be taken into consideration.

DESIGNING WITH THE 49C466

As typical application examples, we consider 32-bit and 64-bit processor designs using EDC to protect the slower dynamic main memory. On account of its 64-bit data buses, the 49C466 is easily interfaced to 64-bit systems. As mentioned earlier, providing a 64 bit memory subsystem bus is often advantageous even in 32 bit processor systems. Thus, even though the interface between 32- and 64-bit wide buses may be slightly more complex, it is often worthwhile. Consequently providing a 64-bit EDC may still be an attractive choice.

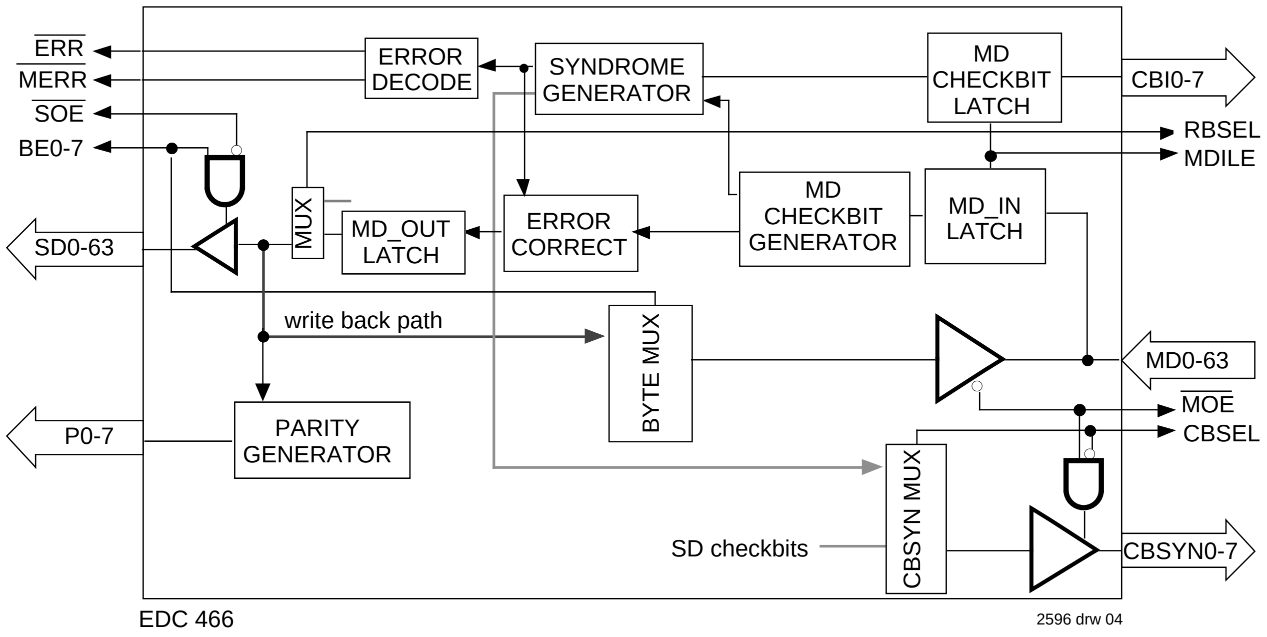


Figure 4. Memory Read without Read Buffer

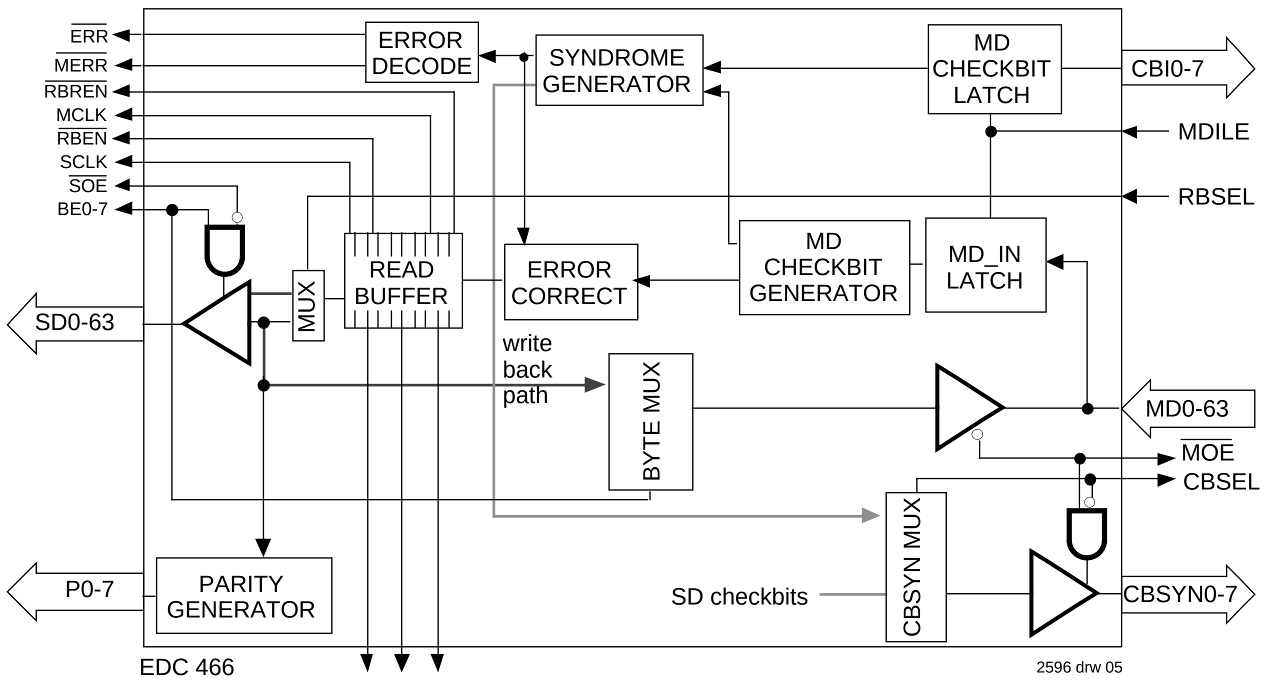


Figure 5. Memory Read with Read Buffer

R3000 BASED SYSTEM

The MIPS R3000 RISC architecture has attained much popularity in recent years. Therefore, we discuss here a design example showing the interface between the 32-bit R3000 and the 64-bit 49C466. This typical R3000 design employs 2 data buses. The high speed “processor” data bus is the interface between processor, caches & buffers while the slower ‘memory’ data bus links the buffers and I/O devices (peripherals & memory). Providing two separate buses for the processor and memory subsystems, in this manner, serves multiple purposes. It reduces the traffic on each bus and allows for different speeds and widths on the two buses.

In the design shown, the 49C466 is positioned between the 64-bit memory bus and main memory as in Figure 6. Programmable logic is used to generate the control signals for the 49C466 64-bit flowthruEDC™. Appropriate control signals from the R3000 processor provide input to the PAL.

The example considered here does not use the on-chip read/write buffers of the 49C466 and the device is used in a non-pipelined mode. External read/write buffers are provided as the interface between the 32-bit processor bus and the 64-bit memory bus. A 64-bit wide memory bus demands that an appropriate scheme for (32-bit) word gathering be devised. One method of implementing such a scheme is described here.

On write cycles, two 32-bit words are gathered in the two external write buffers and the 64-bit word is output to the memory bus.

Gathering 32-bits words, in this fashion, may not always be permissible because of requirements that the two 32-bit words be written to specific, non-sequential locations. Such cases are treated as partial word writes by the EDC. Each 32-bit word write is handled as an individual partial word write cycle. A partial word write cycle requires that the 32-bit word from the adjoining memory location is read before the whole 64-bit word is written back to memory.

The two categories of writes - “gathering permissible” and “gathering not permissible” - are differentiated by comparing the write addresses generated by the processor for the two words. This demands that the first write address must be latched until the next write request is received. If the two addresses are not consecutive, a partial word write signal is sent to the PAL. This initiates a partial word write cycle for writing the first word to the EDC.

To understand this particular situation better, let us consider a case where two write addresses have been compared and found to be non-sequential. The first write data is buffered in WBufA and the second one in WBufB. Each of these write request are treated as “partial word writes” and handled in the order they were received. In order to support byte access, the memory must be byte addressable. The contents of

WBufA are output to the SD bus of the 49C466. To perform the “partial word write” cycle contents of the location address output from WBufA are simultaneously read

onto the MD bus of the 49C466. Merging of these two 64-bit wide data words is performed inside the 49C466 and the merged 64-bit word is output on the 49C466 MD bus and written back to the same location. When this cycle is complete, the same thing is done for the second write data which is stored in WBufB.

During read cycles, the issue of gathering “reads” and comparing read addresses can be avoided altogether by always reading a 64-bit word from memory.The required 32 bits are selected and the rest ignored.

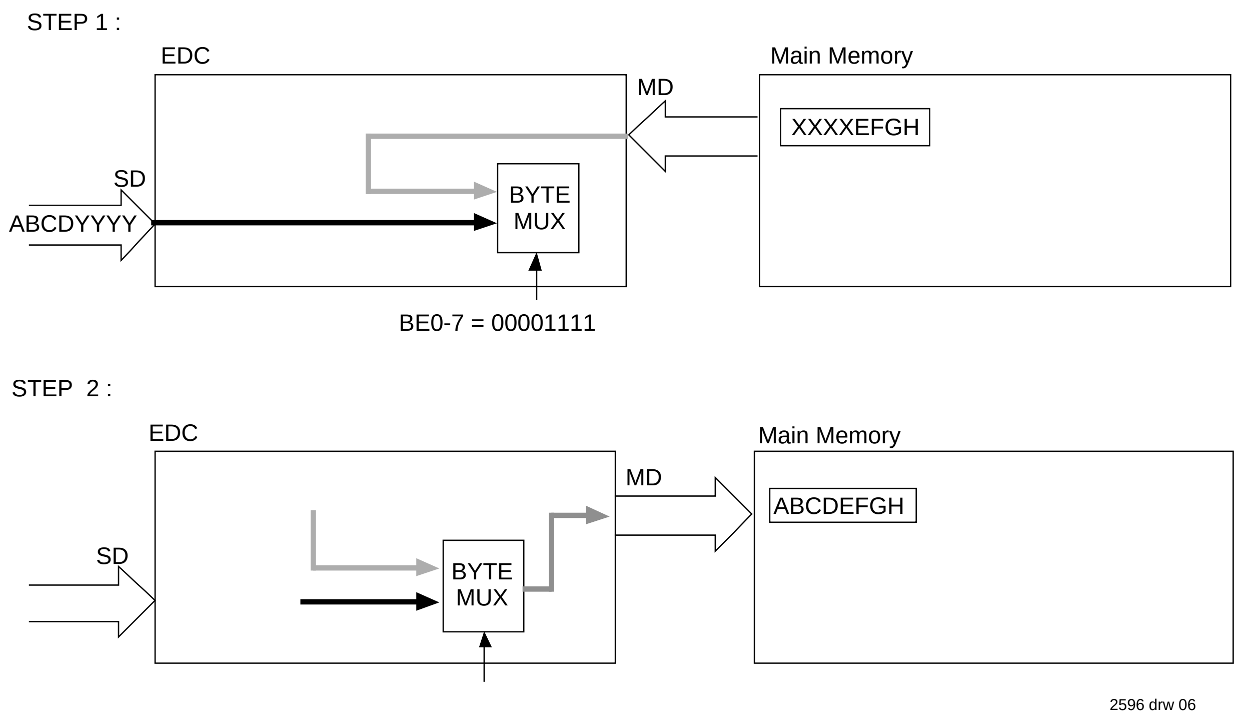


Figure 6. Byte Merge

Details of this scheme, implemented with IDT parts, are shown in Figures 7 and 8.

The design shown, uses a bus multiplexer, the IDT49FCT804 to transfer each 32-bit word to the appropriate SD bus lines (0-31 or 32-63). The write buffer used (IDT 79R3020) buffers both addresses and data. Two bus multiplexers are used to create a crossbar type of arrangement, so that the output of each write buffer can be directed to either the upper or lower SD lines of the 49C466.

64 BIT SYSTEMS

The 49C466 interface to a 64-bit processor is quite straightforward. Intel's i860 and MIPS' R4000 are two popular processors matching this buswidth. Figure 9 shows an i860 based system with EDC. In the system shown all memory accesses go through the 49C466. This kind of a setup would ensure filtering of errors from all memory data accessed. The i860 does not support parity but external parity support circuits (like the AMD280) enable the user to still take advantage of the parity feature of the 49C466. This is particularly useful when caches are employed and some monitoring of data written to main memory is required.

As mentioned earlier, with the trend to segregate processor and memory subsystem designs, a 64 bit memory bus is not unlikely in a 32-bit processor design. In these cases too the 49C466 interface is similar to the one shown above.

MORE EFFECTIVE EDCS

Most present generation EDC units are able to correct only single bit errors. Given the probability of occurrence of multiple bit errors, this is, in most cases, sufficient. The ratio of single bit errors to dual bit errors ranges from 5:1 to 10:1. The probability of multiple bit soft errors occurring is negligible. Also additional hardware is required to correct more than one bit in error. Taking these factors into account, little motivation to move towards multiple bit error correction can exist at this point.

Since hard errors are non-random it is possible for hardware to differentiate them from soft errors. Once this is done it is a simple matter to correct the data provided there is not more than one soft error present. Given below is a brief description of how hard errors can be eliminated.

When an error occurs, the error data is latched in the "error data register" of the 49C466. This data can be read by the processor in the Error Data Mode. In order to filter out hard errors, this data should be inverted and written back to the same location. A subsequent read will serve to identify hard errors, for due to the hard error, the affected bits remain at their original logic level.

Consider a case where bits 3,6 and 7 (shown by bold letters) are affected by hard errors:

Correct Data	= 10101001
Error Data (ED)	= **01**1**0**0001
Inverted error data	= 10011110
Inverted error data after subsequent read (SIED)	= 01010110
By XOR-ing the Error Data with the data from the subsequent read, the bits in (hard) error are isolated (indicated by zeroes). (SIED) XOR (ED)	= 00110111

Thus, all that is externally needed to perform this check is XOR logic.

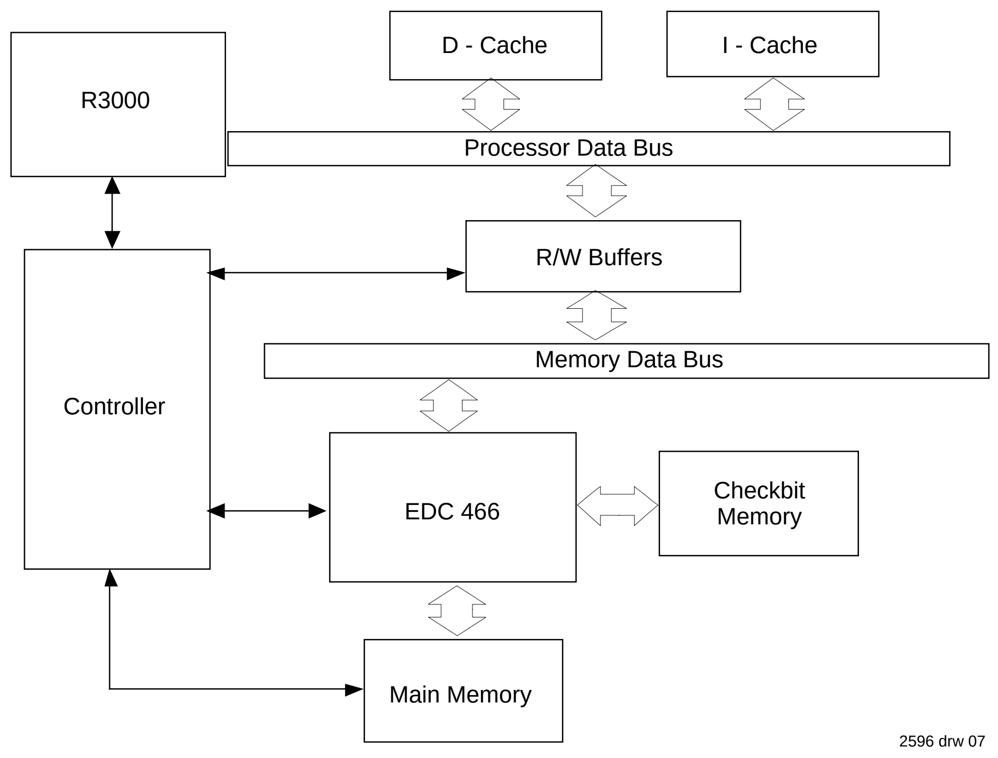


Figure 7. R3000 based System with EDC

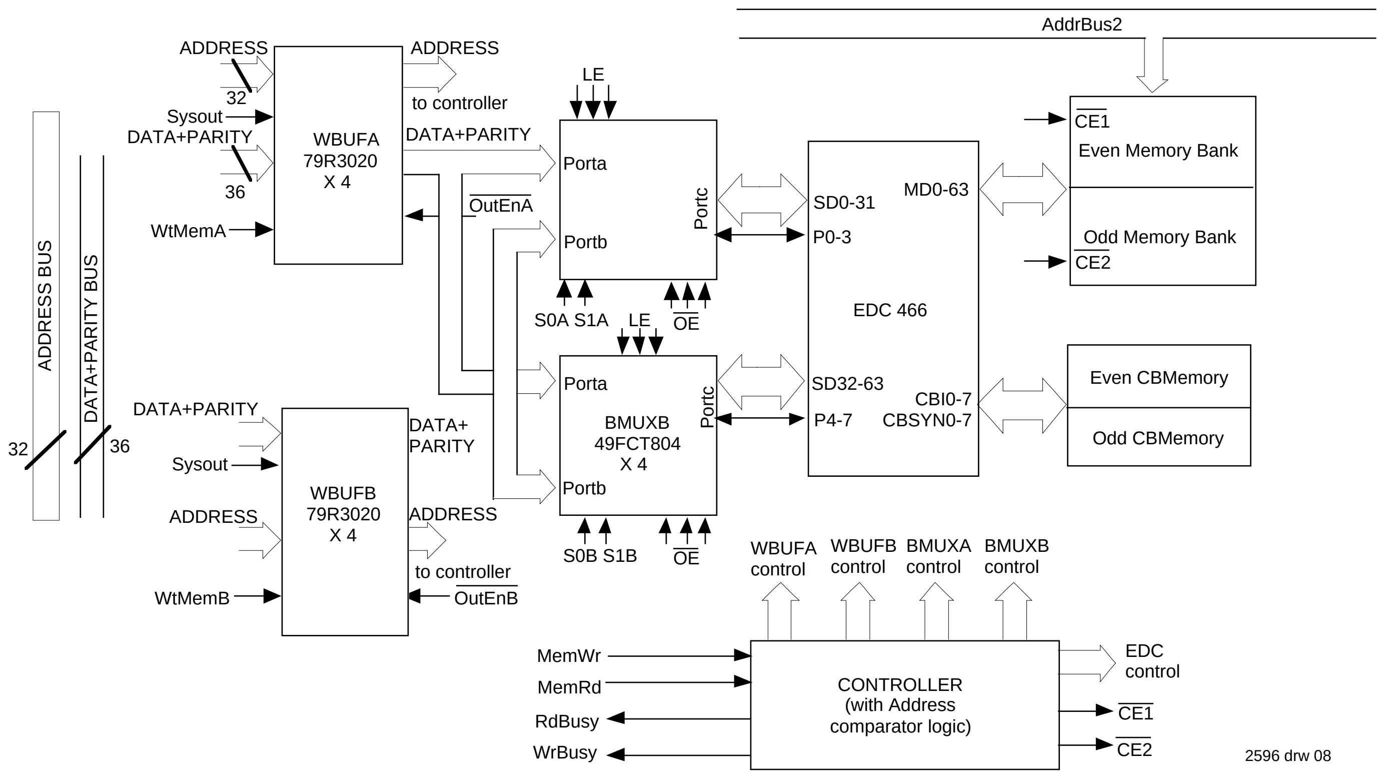


Figure 8. R3000 based EDC System - Memory Write

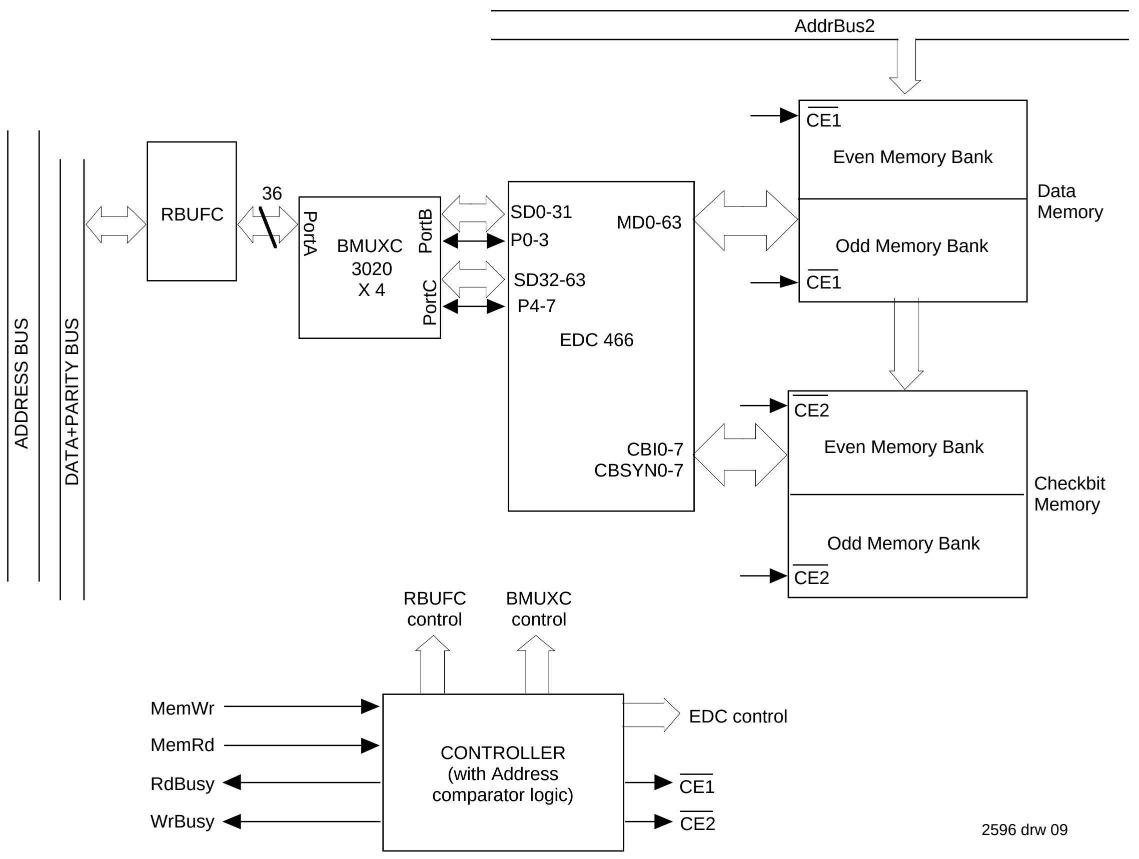


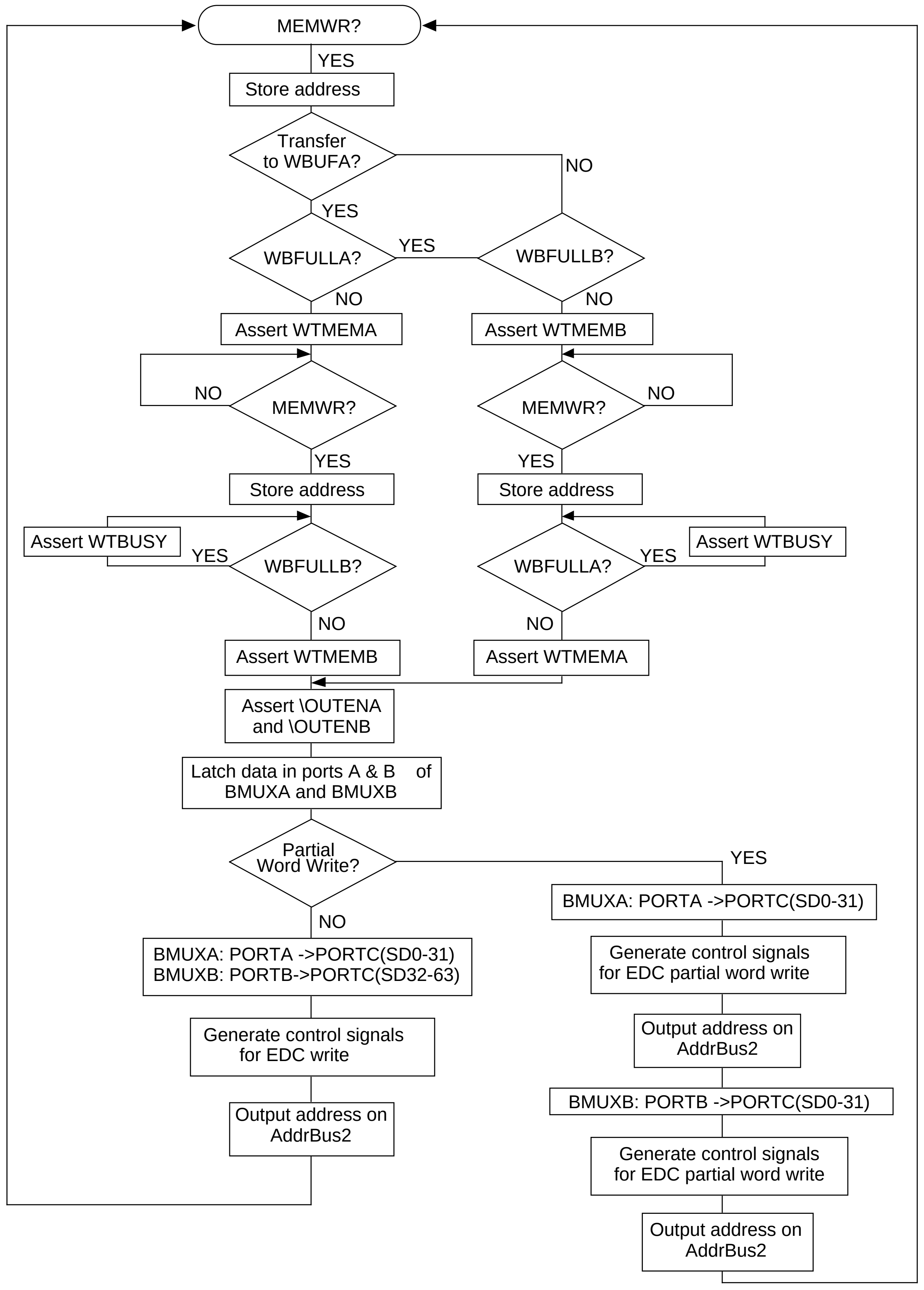
Figure 9. R3000 based EDC System - Memory Read

MEMORY SCRUBBING

An alternative (or in addition) to putting the EDC directly in the read/write path, is putting it in the DRAM refresh path. This ensures that memory is periodically checked for data validity. This is a good practice as it prevents buildup of single bit errors at infrequently accessed memory locations. There are, however, chances of errors slipping through the net in this scheme when an error occurs between a memory refresh and the next memory read or write.

CONCLUSIONS

The urge to provide greater value and reliability in today's competitive computer systems, is likely to make EDC a standard feature in the near future. With the continuing trend towards wider buses, it is only natural to move towards wider EDCs. This also makes the EDC implementation more efficient. IDT is at the forefront of this new generation of EDCs with the powerful new 64-bit flowthruEDC, the 49C466. The 49C466 has an edge over it's competition by providing several useful features such as byte merge capability, 16 deep buffering, latches, diagnostic registers, parity generation and checking and competitive detect and correct times. Currently the device is available in 208 pin PGA and PQFP packages.


MEMWR?
YES
Store address
Transfer to WBUFA?
NO
YES
WBFULLA?
YES
WBFULLB?
NO
NO
Assert WTMEMA
Assert WTMEMB
NO
MEMWR?
MEMWR?
NO
YES
YES
Store address
Store address
Assert WTBUSY
YES
WBFULLB?
WBFULLA?
YES
Assert WTBUSY
NO
NO
Assert WTMEMB
Assert WTMEMA
Assert \OUTENA and \OUTENB
Latch data in ports A & B of BMUXA and BMUXB
Partial Word Write?
YES
NO
BMUXA: PORTA ->PORTC(SD0-31)
BMUXA: PORTA ->PORTC(SD0-31)
BMUXB: PORTB->PORTC(SD32-63)
Generate control signals for EDC partial word write
Generate control signals for EDC write
Output address on AddrBus2
Output address on AddrBus2
BMUXB: PORTB ->PORTC(SD0-31)
Generate control signals for EDC partial word write
Output address on AddrBus2

2596 drw 10

Figure 10. Flowchart for Write Buffer and Bus Multiplexer Control During Memory Write

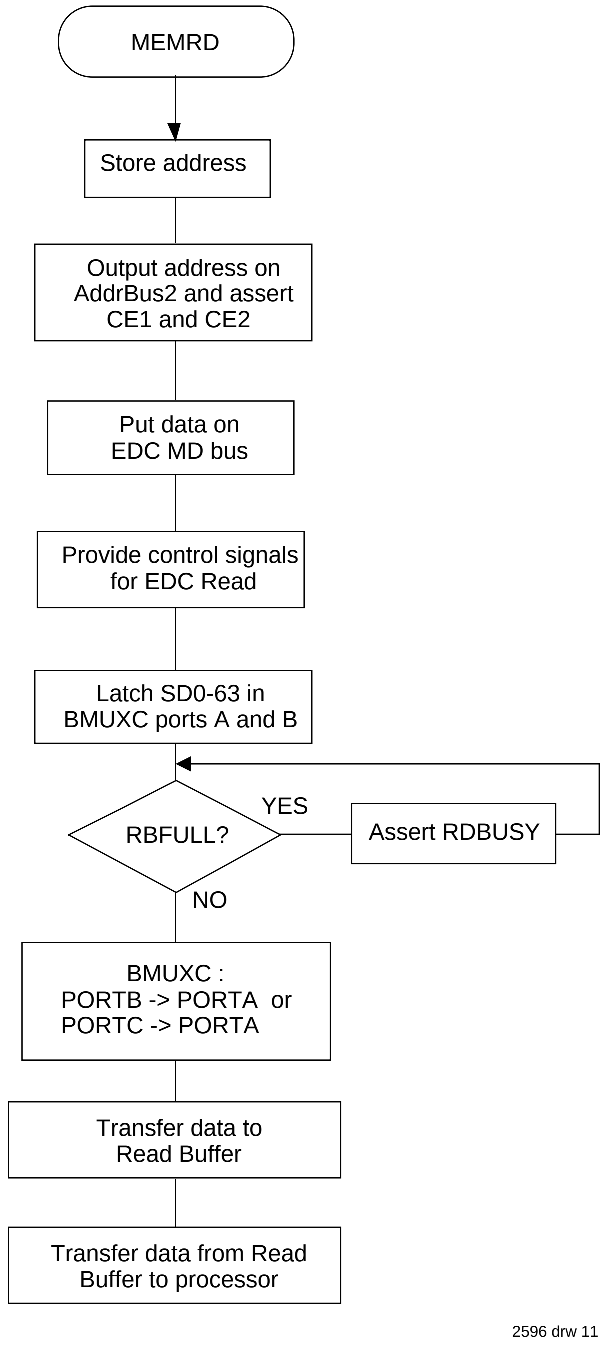

MEMRD
Store address
Output address on AddrBus2 and assert CE1 and CE2
Put data on EDC MD bus
Provide control signals for EDC Read
Latch SD0-63 in BMUXC ports A and B
RBFULL?
YES
Assert RDBUSY
NO
BMUXC : PORTB -> PORTA or PORTC -> PORTA
Transfer data to Read Buffer
Transfer data from Read Buffer to processor
2596 drw 11

Figure 11. Flowchart for Read Buffer and Bus Multiplexer Control During Memory Read

APPENDIX

MTBF Calculations

DRAM manufacturers specify a soft error rates in terms of FITs (failures in time). Assume a 1M x 1 DRAM with soft error rate = 252 FITs

Thus,

$$\text{MTBF for each 1M x 1 DRAM chip} = \frac{10^9}{252} = 453 \text{ years}$$

$$\text{MTBF for a 4M x 64 memory system (without EDC)} = \frac{453}{4 \times 64} = 1.76 \text{ years}$$

[A memory system without EDC assumes a system failure each time a single bit error occurs. This may not be the reality but this exercise is aimed at showing the difference between two analogous cases, hence such assumptions are in place. Failures due to all higher order (dual, three bit, etc.) errors can be safely ignored in this case due to the overwhelming dominance of single bit errors over other higher order errors.]

With EDC, extra memory is required to store checkbits, hence the number of DRAM chips required for the same memory system goes up.

$$\text{Checkbit memory (with 64 bit EDC) required for the above system} = \frac{1}{8} \times 4 \times 64 = 32 \text{ chips}$$

Total memory system chip count = (4 x 64) + 32 = 288

So now,

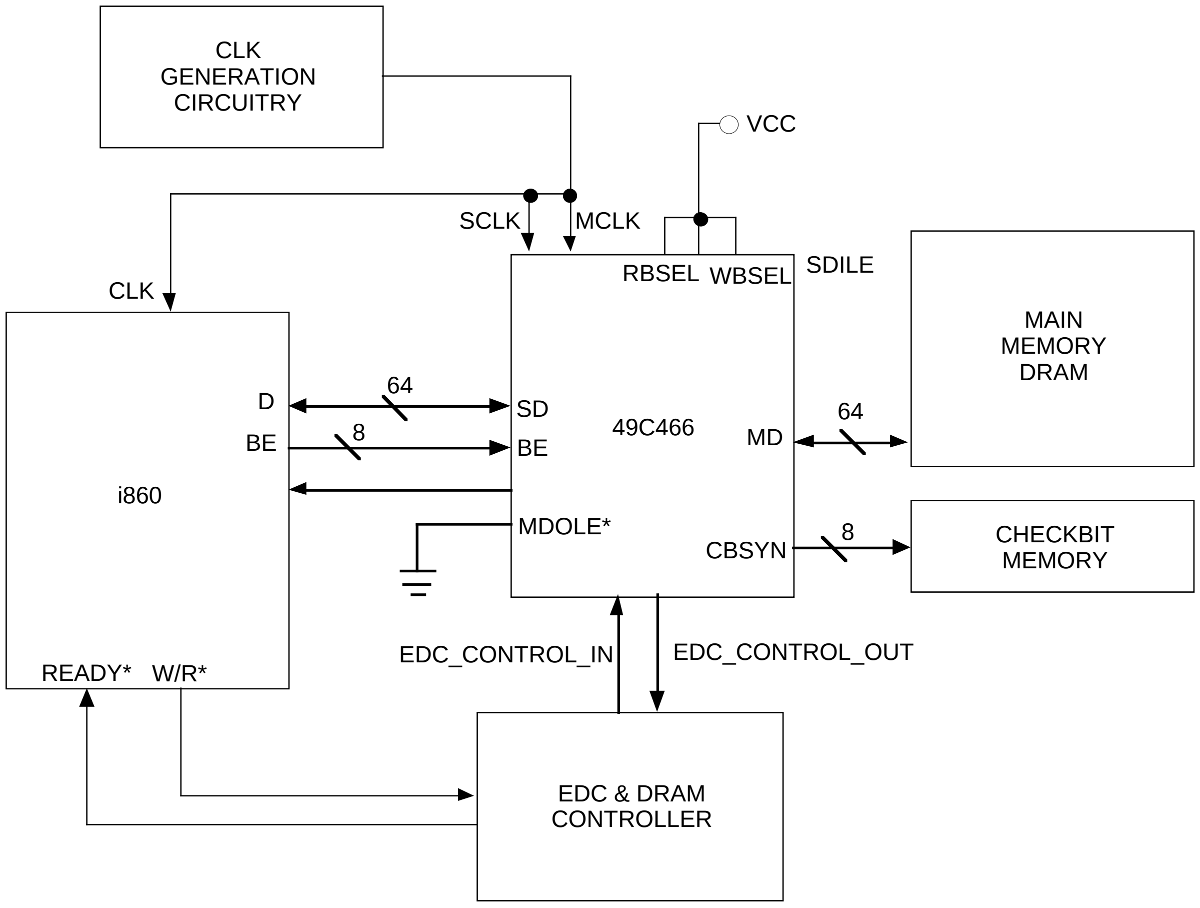
$$\text{Memory system MTBF} = \frac{453}{288} = 1.57 \text{ years}$$

Now if we assume that with EDC, all single bit errors are corrected, failures due to dual bit errors become dominant. Neglecting failures due to higher order errors, we get an approximation of the MTBF using the following formula :

$$\text{MTBF (with EDC)} = (\text{MTBF without EDC}) \times \sqrt{\left(\frac{\pi \times \text{\# of memory words}}{2}\right)}$$

$$= 1.57 \times \sqrt{\left(\frac{\pi \times 4 \times 10^{12}}{2}\right)} = 3935.4 \text{ years}$$

Thus we see that there is a significant improvement in the MTBF of a system with EDC.



NB : EDC_CONTROL_IN : SDOLE*,MDILE,WBEN*,WBREN*,RBEN*,WBREN*,CBSEL,RS0-1,

EDC_CONTROL_OUT : WBEF*,WBFF*,RBEF*,RBFF*,RBHF*,ERR*,MERR*

2596 drw 12

Figure 10. Flowchart for Write Buffer and Bus Multiplexer Control During Memory Write