Controlador de Memória Dinâmica para o μP68030

N. Borges Carvalho, R. Vieira Silva e A. Nunes Cruz

Resumo **- Apresenta-se o projecto dum controlador, que implementa a interface do μP68030 a uma memória dinâmica. A memória permite o acesso de bytes, words e longwords, suportando ainda um modo de acesso rápido ('Nibble'). Para aumentar a confiabilidade ('reliability'), a interface inclui ainda um circuito de detecção e correcção de erros.**

I. INTRODUÇÃO

Pretende-se interligar uma memória dinâmica (DRAM) e o μProcessador MC68030 da Motorola. Antes de gerar uma especificação detalhada do controlador, é apresentado duma forma progressiva e pedagógica, o background necessário ao seu entendimento.

Dum lado temos a memória, do outro o μP68030. No tocante à memória, começamos por abordar o aspecto da sua estrutura interna, descrevendo os vários tipos de acesso que a mesma permite, com destaque para o modo de acesso rápido do tipo 'Nibble'. As várias modalidades de 'refresh' são também descritas. Do lado do CPU, é explicado o funcionamento da respectiva interface ao bus.

Em seguida, é apresentada a especificação do controlador, através da descrição funcional dos vários blocos, do tipo de ciclos de acesso suportados e ainda dos diferentes modos de refresh possíveis.

Finalmente, é apresentada uma solução para a detecção e correcção de erros através do ciclo de 'scrubbing *refresh*' das memórias, durante o qual o conteúdo da DRAM poderá ser corrigido. A inclusão desta facilidade no controlador, é também analisada.

Na fig.1 apresenta-se um diagrama de blocos total do sistema final. É constituído pelo microprocessador MC68030, o controlador por nós projectado, pela memória DRAM de armazenamento de dados e ainda por um bloco de memória extra, na qual se armazena a informação redundante (os '*check-bits*') necessária à detecção e correcção de erros e finalmente pelo circuito EDAC ('Error Detection And Correction'), o qual implementa o algoritmo de Hamming que possibilita a correcção de um bit numa palavra de 32-bits.

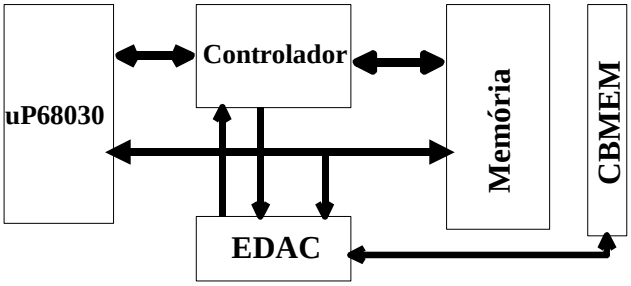

Figura 1 - Diagrama de blocos total.

II. ACESSO À MEMÓRIA DINÂMICA

A. *O Chip de memória Dinâmica*

A memória dinâmica (TMS4C1025) a utilizar é do tipo da apresentada na figura seguinte:

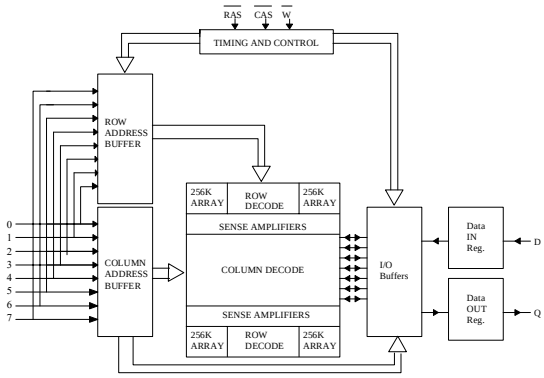

Figura 2 - Diagrama de blocos da memória (1Mx1Bit).

Os blocos que a compõem são:

- *Timing and Control*: Gera todos os sinais de temporização e controlo, necessários aos diversos modos de acesso e ainda ao *refresh CAS-before-RAS* .
- *Row address buffer*: *Buffers* do endereço de linha.
- *Column address buffer*: *Buffers* do endereço de coluna.
- 256k *Array*: Os 4 blocos de 256Kx1bit, da memória de 1Mbit.
- *Column Decode*: Descodificador de coluna.
- *I/O buffers*: Além de buffers bidireccionais, servem para fazer acessos em modo *Nibble* (ver acesso Nibble).

Os sinais de entrada e saída da memória são:

- 0-7 : *Address*, é o bus de endereços.
- W : *Write*, é o sinal que indica à memória a operação de leitura ou escrita.
- CAS : *Column Adress Strobe*, este sinal informa a memória de que o endereço presente nas linhas 0-7, selecciona a coluna da célula de memória a aceder.
- RAS : *Row Adress Strobe*, informa a memória de que o endereço presente nas linhas 0-7, selecciona a linha da célula de memória a aceder.
- D : Dados, é o bus de dados de entrada.
- Q : Dados , é o bus de saída de dados.

B. *Ciclos de Acesso Básicos*

B1. Leitura

O ciclo de leitura começa quando o controlador coloca o endereço de linha em A0..A9 e põe a linha de RAS a *low* fazendo o *strobe* dos endereços para a DRAM. O RAS terá que se manter a *low* durante todo o ciclo. A linha de CAS não só informa a memória do envio do endereço de coluna como também activa os *buffers* de saída e portanto, deve-se manter a *high* enquanto se faz o *strobe* do endereço de linha para a DRAM.

Em seguida, o controlador envia o endereço de coluna para a DRAM e após a estabilidade deste, coloca a linha de CAS a low validando a coluna. Enquanto o CAS se encontra em low, a linha do WRITE deve manter-se high de maneira a activar o ciclo de leitura.

Para garantir que a informação de saída seja válida, deverá ter decorrido um tempo mínimo ta(R) depois da descida da linha de RAS, e ta(C) depois da descida de CAS.

Para terminar o ciclo de leitura, o controlador terá que por as linhas de RAS e CAS a high. Mesmo assim, o ciclo não se completa. O próximo ciclo não poderá ser iniciado até que a DRAM, durante um período de *'precharge'*, escreva novamente e de forma automática a informação previamente acedida (Fig.3).

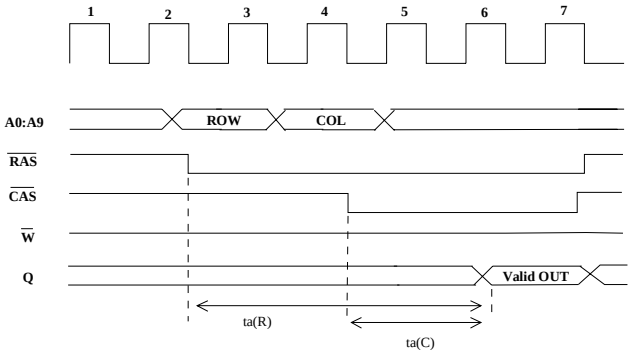

Figura 3 - Ciclo de Leitura.

B2. Escrita

A escrita para a DRAM envolve a maioria dos mesmos parâmetros do timing dum ciclo de leitura, caso concreto do RAS, CAS e A0..A9. A principal diferença está na linha de WRITE que envia a informação para a memória dinâmica depois do controlador colocar a linha de CAS a low (Ciclo de Delayed-Write). Durante o ciclo de escrita a informação deve encontrar-se estável antes da descida do WRITE e manter-se assim durante um período especifico de tempo (Fig.4).

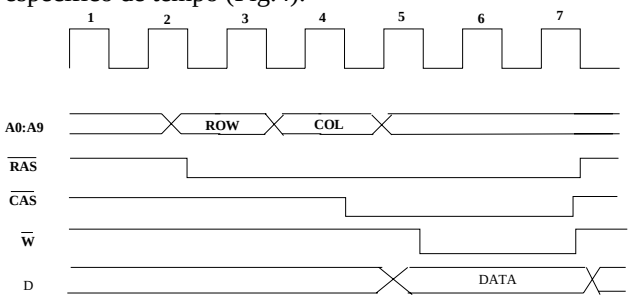

Figura 4 - Ciclo de Escrita.

C. Ciclo de 'Read-Modify-Write'

Para o ciclo do 'Read-Modify-Write', os parâmetros de timing são uma composição dos ciclos de leitura e escrita. A linha do WRITE é posta a high antes da descida do CAS mas não se mantém a high durante todo o tempo que o CAS se encontra a low. As linhas de Data-In contêm agora informação válida, as quais assim se manterão durante um tempo específico de hold após a descida do WRITE. A linha do WRITE desce para low validando a informação nas linhas de entrada (Fig. 5).

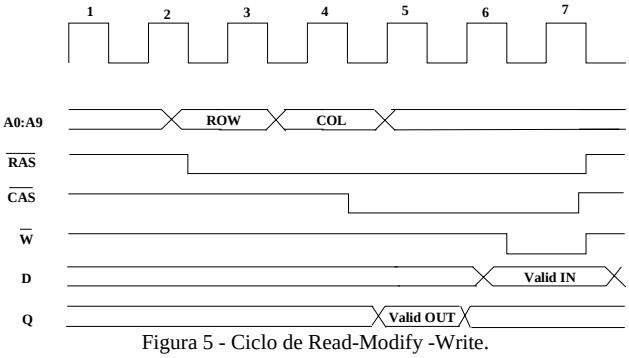

Figura 5 - Ciclo de Read-Modify -Write.

D. Acesso Nibble

Operações em modo Nibble permitem acessos de leitura, escrita ou 'Read-Modify-Write' de 1 a 4 bits de data. Este modo pode ser facilmente compreendido analisando o seguinte diagrama temporal

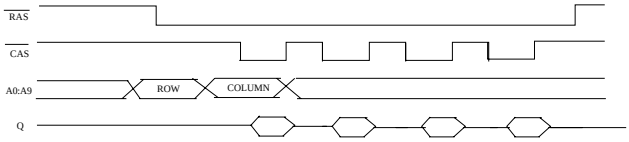

Figura 6 - Diagrama temporal de um acesso *Nibble*.

O acesso ao primeiro bit é um acesso normal, em que a memória consegue aceder à informação 25ns após a descida do CAS. Os bits seguintes podem ser lidos ou escritos sequencialmente, comutando o CAS sucessivamente enquanto se mantém o RAS a low.

A memória estará dividida em 4 blocos de 256Kbits. Os 20 bits, necessários para endereçar internamente 1M células de 1-bit, são obtidos a partir dos 10 bits do endereço de linha (pelo latching do RAS), e dos 10 bits do endereço de coluna (pelo latching do CAS). Durante o primeiro acesso, o valor do bit A9 (tanto do endereço de linha como do da coluna) determina qual dos 4 blocos de memória é o primeiro a ser acedido. Após este acesso inicial, os sinais de endereço externos não são mais usados. Os 3 próximos endereços sequenciais, são gerados por lógica interna da DRAM, a partir da comutação do CAS. É importante notar que estes 4 bits ('Nibble'), são obtidos internamente - 1 de cada bloco de 256k- durante o primeiro acesso, sendo disponibilizados para o exterior, duma forma que corresponde ao endereçamento circular dos vários blocos internos de 256k (Fig. 7).

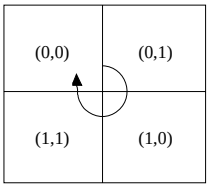

Figura 7 - 4 arrays de 256k acedidos em modo *Nibble* através da comutação do CAS.

O modo Nibble é mais rápido do que o modo Pagina, (embora este ultimo modo permita acessos aleatórios a uma linha e uma coluna), basicamente porque o modo Nibble permitir aceder a 4 bits com um único endereço de linha e coluna, ao passo que no modo Página para aceder ao bit seguinte da mesma página (linha) a memória precisa, também, de descodificar o respectivo endereço de coluna.

E.Ciclos de Refresh

'Refresh' é a operação que consiste em revitalizar a carga de cada célula de memória, a qual de outro modo diminuiria, levando à perda da informação armazenada. Portanto, a tarefa mais prioritária do controlador de DRAM é garantir que esta realize um 'refresh', de todas as células, dentro de um período máximo de 8ms.

Para desencadear um ciclo de 'Refresh' da DRAM, basta que o controlador seleccione uma das suas linhas. Esta operação tanto pode ocorrer durante um ciclo especifico de refresh (e.g. 'RAS Only'), como durante um ciclo normal de acesso leitura/escrita duma célula. Neste último caso, para além da célula acedida, também as demais células pertencentes à mesma linha, são refrescadas.

Essencialmente, os modos mais comuns de fazer o refresh duma DRAM, distinguem-se segundo a localização (externa ou interna à DRAM) do contador que gera os endereços de linha.

E.1. 'Ras-Only'

Um dos métodos de refresh é o 'Ras-Only'. Neste caso, os endereços de linha da DRAM a refrescar são fornecidos exteriormente e validados pela linha de RAS enquanto a linha de CAS é mantida a high pelo controlador. Enquanto a linha de RAS faz o strobe do endereço de row para dentro da DRAM, esta durante o refresh não pode fazer qualquer transferência de data, porque o CAS mantém-se sempre a high, não permitindo o enable dos drivers de saída.

E.2. 'CAS-Before-RAS'

Outro método de refresh automático (suportado pelas memórias usadas neste projecto) é o 'CAS-before-RAS'. Neste caso, o controlador coloca o CAS a low antes de activar o RAS; esta sequência temporal, activa o gerador interno de endereços da própria DRAM. Assim, o próximo o endereço de linha a refrescar é gerado pela própria DRAM e não externamente. Esta característica da DRAM, simplifica o controlador, na medida em que este não necessita de possuir um contador de endereços de refresh.

E.3. 'Scrubbing Refresh'

O método de refresh mais completo é o Scrubbing Refresh; neste ciclo o controlador desencadeia um ciclo de leitura de dados, activa a detecção e correcção de erros e, caso seja necessário, volta a escrever a informação corrigida na DRAM. Para detectar a ocorrência de erros o controlador interactua com a unidade EDAC. Este ciclo será explicado mais em pormenor na secção VI.

III. ORGANIZAÇÃO DE BANCOS DE MEMÓRIA

A. Interligação das Memórias e Acessos

Na estruturação da memória cria-se um esquema de ligação de memórias de 1Mbit de modo a se poder fazer acesso de byte, word ou longword. O acesso byte, apenas equivale a colocar em paralelo 8 memórias de 1Mbit, conseguindo desse modo aceder simultaneamente a 1MByte (Fig. 8). Como também se pretende ter acesso a word e longword, ter-se-á também 2 conjuntos de 1MByte para word e 2 conjuntos de 1MWord para longword. Para obter os 16 Mbytes teremos uma distribuição de memórias como na Fig.9.

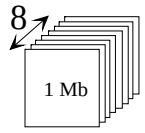

Figura 8 - 1 MByte.

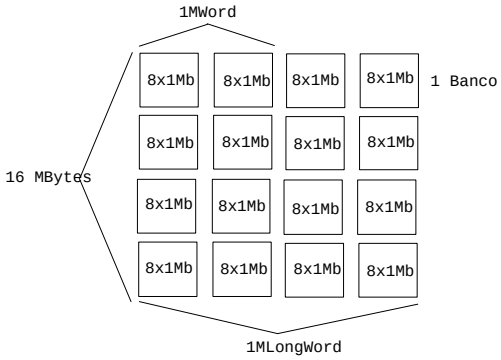

Figura 9 - Organização da memória.

Para aceder às memórias o microprocessador usa o bus de endereços A0..A31. Para aceder a 16 MBytes basta usar as linhas de endereço A0..A23 e não considerar A24..A31. Deste modo é possível considerar vários mapas de memória, que são acedidos usando as mesmas linhas A0..A23, dependendo da descodificação de A24..A31 (Fig. 10).

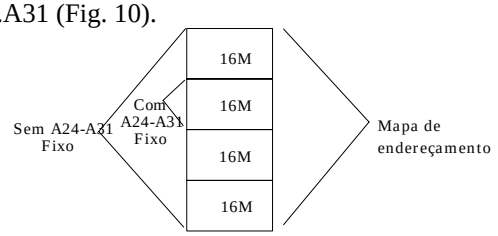

Figura 10 - Mapa de endereçamento.

Uma vez escolhido o método, fixando ou não o A24..A31, dividem-se os restantes endereços A0..A23 de modo a aceder à memória.

Começa-se por usar os bits de endereços A22..A23 para seleccionar qual o banco de longword a aceder na

Fig. 9. RASi/CASi são obtidos, a partir de A22 e A23, com um simples descodificador de 2 para 4.

Porque o microprocessador utiliza os sinais A0, A1, SIZ0 e SIZ1 para agulhar os bytes de entrada/saída, optou-se por usar A0 e A1 em associação com SIZ0 e SIZ1 para seleccionar qual o byte ou word a aceder quando o acesso não é em longword. Como do lado da DRAM, o byte(s) seleccionado(s) é indicado através dos sinais de 'Write' W0..W3, estes últimos são facilmente obtidos em função de A0, A1, SIZ0 e SIZ1.

Sobram A2..A21 para aceder a cada memória de 1Mbyte. Estes endereços são divididos a meio, usando A2..A11 para as linhas e A12..A21 para as colunas, para aceder a cada célula (ou byte se considerarmos 8 unidades de 1Mbit em paralelo) da DRAM, uma vez que esta recebe os endereços duma forma multiplexada.

Usando os sinais de CAS e RAS para escolher o banco e WRITE para escolher o byte, ficamos com as memórias interligadas como na Fig. 11.

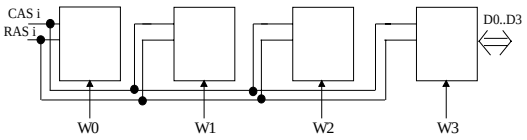

Figura 11 - Interligação de RAS e CAS.

Deste modo, fica definido o acesso lógico às memórias DRAM.

IV. INTERFACE DO CPU AO BUS

Normalmente a organização da memória é feita em bancos com uma largura igual à largura do bus de dados do CPU. A organização da Fig. 11, designada por 'Bank-Oriented-CAS', requer que as linhas de CAS (Column Address Strobe) e RAS (Row Address Strobe) sejam comuns a cada bloco dentro de um banco, sendo a selecção de um único bloco ('byte') feita através de uma linha de WRITE.

O CPU 68030 pode aceder a um byte, word ou longword, o que implica que o controlador necessita de gerar os sinais W0..W3 consoante o tamanho da palavra. Este CPU indica o tamanho e secção válida do bus de dados através dos sinais SIZ0, SIZ1, A0 e A1.

Os bits de dados D0..D31 não entram no controlador da DRAM (apesar deste controlar a sua transferência indirectamente), mas passam ou directamente para a memória, ou através de um circuito EDAC em paralelo com o controlador.

A interface ao bus do 68030 usa um protocolo de transferência com handshake, implementado através dos sinais AS (Address Strobe), DS (Data Strobe) e DSACK0-1 (Data Acknowledge).

A figura seguinte ilustra os sinais gerados pelo CPU e os correspondentes sinais do controlador para executar um acesso típico à DRAM.

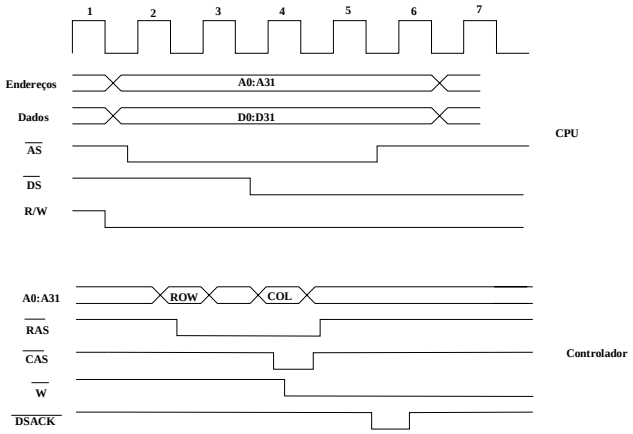

Figura 12 - Acesso de escrita na memória.

A. Sinais do CPU

Address Bus de 32 bits usado para o endereçamento até 4GB.	A0..A31
Data Bus usado para fazer transferências de 8, 16 ou 32 bits.	D0..D31
Sinais que indicam o numero de bytes que faltam transferir num ciclo. Juntamente com A0 e A1 definem as secções activas do bus.	SIZ0/SIZ1
Read/Write	R/W
Ciclo de Read-Modify-Write.	RMC
Address Strobe.	AS
Data Strobe.	DS
Indicam que o pedido de transferência está completo e qual o tamanho do port.	DSACK0/ DSACK1
Indica um pedido de Burst.	CBREQ
Indica que o dispositivo acedido pode operar em modo Burst.	CBACK
Indica acesso ou operação inválida no bus	BERR

V. ESPECIFICAÇÃO DO CONTROLADOR

Da exposição feita, poderemos concluir que a especificação do controlador deverá compreender os seguintes elementos funcionais:

- *Interface ao CPU* - interpreta os sinais provenientes do CPU (R/W, AS e DS) e gera os sinais de handshake (DSACK), por forma a satisfazer os protocolos de transferência.

- *Interface à DRAM* -composta por:
  Multiplexer de Endereços - multiplexa sobre as linhas A9-A0 da DRAM, os endereços de Linha e Coluna provenientes do CPU.
  Lógica de Control - que gera os sinais de selecção e controlo da DRAM, CASi, RASi e Wi.

- *Refresh Timer* - gerador periódico de pedidos de Refresh.

- *Contador de Refresh* - gerador de endereços de Linha (e Coluna no caso de Scrubbing) necessários ao Refresh.
- *Circuito de Arbitragem*- atribui o recurso único (DRAM), ora ao pedido de maior prioridade proveniente do ‘ Refresh Timer’, ora ao CPU, possibilitando que os acessos (assíncronos um relativamente ao outro) se possam processar sem colisões.

A. *Diagrama de blocos do controlador*

O Diagrama de Blocos e respectivo Diagrama de Estados, propostos para implementar as diversas funções do controlador da DRAM, estão apresentados, respectivamente, nas Figs. 13 e 14.

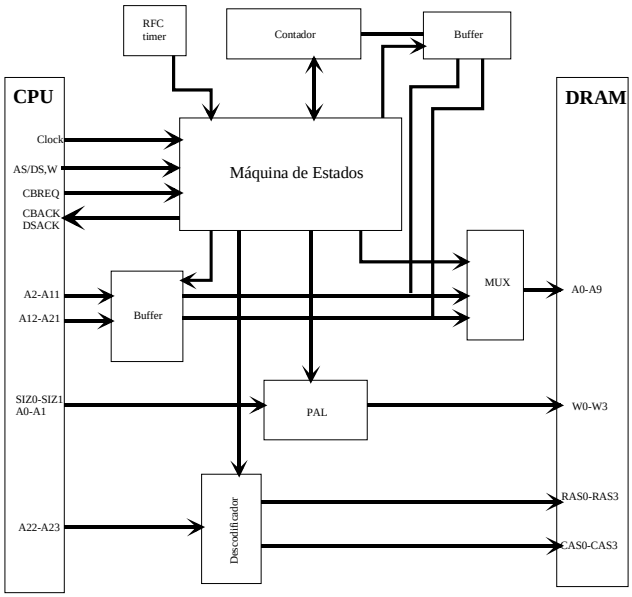

Figure 13- Diagrama de Blocos do Controlador

Neste diagrama merece especial referência a Máquina de Estados. Esta máquina sequencial, cujo diagrama está apresentado na figura 14, é o verdadeiro ‘coração’ do controlador. Arbitra entre os pedidos (prioritários) de refresh provenientes do timer RFC e os pedidos de acesso à DRAM por parte da interface ao CPU. Gera todos os sinais de controlo, necessários, tanto ao correcto funcionamento dos blocos internos anteriormente descritos, bem como da interface à DRAM. O Diagrama de Estados compreende essencialmente os ciclos de leitura, escrita e os ciclos de refresh. Uma descrição mais detalhada dos vários sub-ciclos é apresentada, mais abaixo no ponto D.

B. Diagrama de estados total

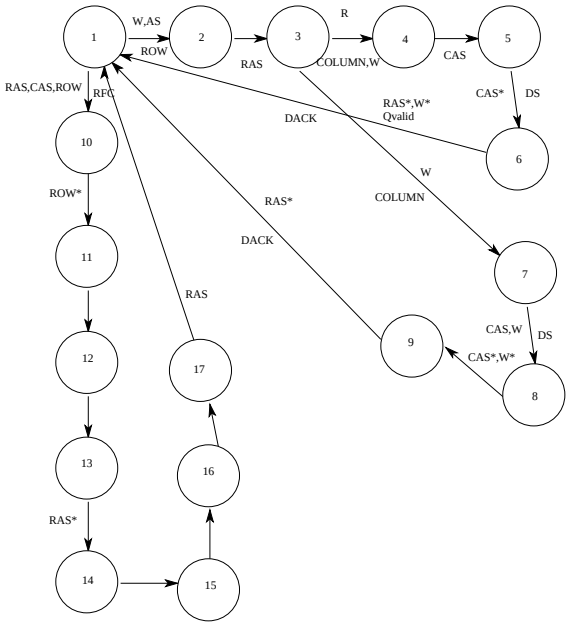

Figure 14 - Diagrama de Estados Total

C. Sinais de Interface à DRAM

Fornecem os endereços de Coluna e Linha.	A0-A9
Strobes de linha para selecção de bancos de memória.	RAS0-RAS3
Strobes de coluna para selecção de bancos de memória.	CAS0-CAS3
Strobes de leitura/escrita para selecção de memórias dentro dos bancos.	W0-W3

D. Diagramas de Estado e Temporais

Apresenta-se, em seguida, uma descrição mais pormenorizada dos vários sub-ciclos, implementados pelo controlador de DRAM.

D.1 Diagrama de estados para o acesso básico de leitura e escrita

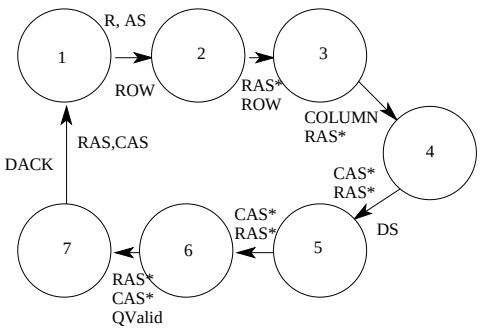

Figura 15 - Diagrama de estados de leitura.

D.2 Diagrama temporal para o acesso básico de leitura

Escolhendo um clock de 40 MHz, obtém-se um tempo de clock de aproximadamente 25ns para a máquina de estados e o diagrama temporal de leitura da Figura 16.

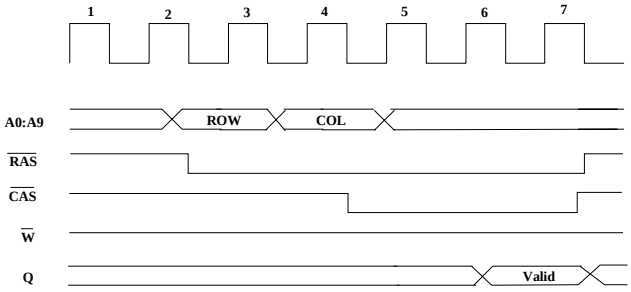

Figura 16 - Diagrama temporal de leitura.

D.3 Acesso em modo Nibble

Tanto a memória como o microprocessador 68030 têm hipótese de fazer transferência em modo Nibble. Este modo é mais rápido do que outros tipos de acesso, pois quando se acede à memória, ele permite ler 4 posições sequencialmente.

Depois de ser feito o acesso normal com o endereço de coluna e linha, basta fazer o toggling do CAS três vezes, de modo a que a memória incremente um contador interno que coloca no bus de dados as posições de memória para cada quadrante.

Os bits de RA9 e CA9 fazem o preset do contador interno, cujo valor selecciona o 1º quadrante (bloco de 256k-bit) a ser acedido. A seguir, com cada comutação do CAS, o contador é incrementado e o multiplexer da memória coloca na saída o bit seguinte. A memória faz assim, o chamado 'Bit-Wrap-Around', uma vez que independentemente do endereço do quadrante inicialmente escolhido, ela coloca cá fora todos os bits dos vários quadrantes (ver fig. 7).

Portanto para fazer um acesso em modo Nibble, o controlador deverá, depois de receber a sinalização do CPU (CBREQ), fazer um acesso normal à memória enviando o endereço de linha, seguido do de coluna e a seguir de acordo com um timing adequado fazer o toggling do CAS 3 vezes mantendo o RAS, de modo a que leia as três unidades seguintes (bits no caso dum chip ou words do caso de vários chips em paralelo), devendo, finalmente, enviar para o CPU o sinal CBACK, para sinalizar a terminação de transferência.

O diagrama de estados e temporal estão identificados nas Figuras 17 e 18.

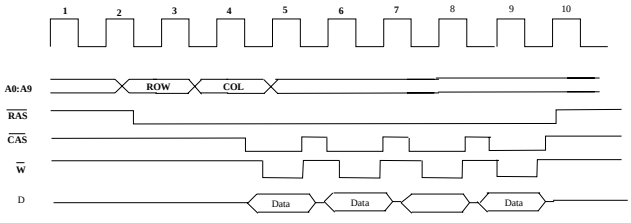

Figura 17 - Diagrama temporal Write *Nibble*.

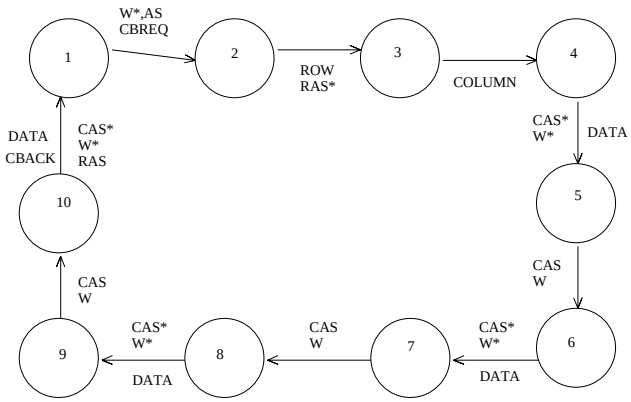

Figura 18 - Diagrama de estados de escrita *Nibble*

E. Ciclo de refresh CAS-before-RAS

O método de CAS-before-RAS é implementado, fazendo com que o sinal de CAS desça antes do sinal de RAS. Assim, quando o controlador receber o sinal vindo do RFC(Refresh Clock), accionará o CAS e após algum tempo, o RAS. O diagrama temporal e de estados são apresentados na Figura 19 e 20.

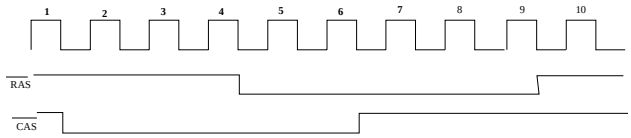

Figura 19 - Diagrama temporal do refresh CAS-before-RAS.

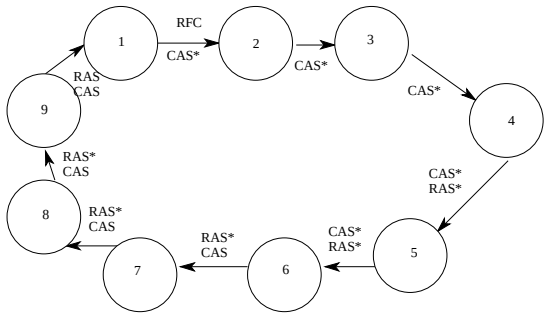

Figura 20 - Diagrama de estados de refresh CAS-before-RAS.

F. Ciclo de refresh RAS-Only

O método de RAS-Only faz o refresh, de uma linha de cada vez, portanto o controlador deverá ter um contador interno que incremente endereço de linha sempre que é feito um refresh. Para o efeito usa-se o RFC, que quando activo incrementa o contador e faz com que o controlador entre num ciclo de refresh.

O diagrama temporal e de estados são apresentados a seguir.

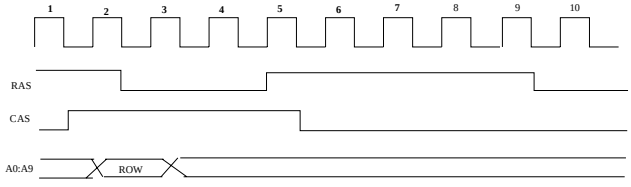

Figura 21 - Diagrama temporal do ciclo de refresh RAS-only.

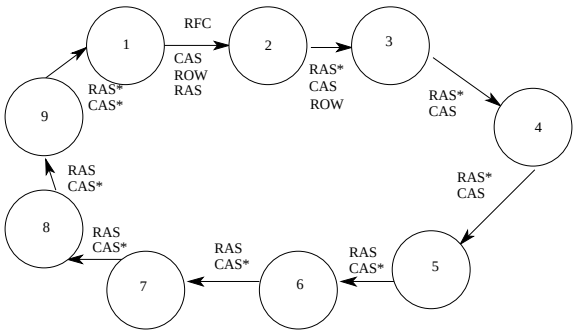

Figura 22 - Diagrama de estados RAS-Only.

VI. DETECÇÃO E CORRECÇÃO DE ERROS

A. Scrubbing Refresh

O ciclo de 'Scrubbing Refresh' é essencialmente um ciclo de refresh do tipo 'RAS-Only' sobreposto a um ciclo de leitura (e escrita caso haja erro) normal. Para efectuar Scrubbing (que significa restauro dos dados armazenados em memória), o controlador necessita essencialmente de duas coisas:

Gerador de Endereços de Refresh - em vez de um simples contador que gere os endereços de linhas, passa a necessitar também de um contador que gere os endereços de colunas.

Circuito de Detecção e Correcção de Erros (EDAC) - este circuito - normalmente externo ao controlador - tem a função de gerar a informação redundante ('checks bits'), que ao ser armazenada numa memória suplementar, possibilita que posteriormente o mesmo circuito possa determinar a ocorrência de erros na informação armazenada na DRAM.

É ao controlador que compete gerar os sinais que controlam o EDAC. Isto é, durante um ciclo normal de escrita, o controlador pede ao EDAC que gere os '*check bits*'. Durante uma operação de leitura, o controlador pede ao EDAC que determine se houve ou não alteração na informação armazenada, e que em caso de erro o informe. Perante o feedback da unidade EDAC o controlador decide se deve desencadear, ou não, uma operação de re-escrita da informação (caso de erro).

Durante um ciclo de Scrubbing Refresh, o controlador gera simultaneamente: um ciclo de leitura normal (Li e Ci) para um banco de memória (e.g. activando RAS0 e CAS0) e um ciclo de 'RAS-Only' (da Linha, Li) para os restantes bancos de memória (activando RAS1-3). Desta forma inteligente, como os endereços de linha correspondem aos menos significativos dos endereços gerados pelo contador de refresh, conseguem-se refrescar todos os bancos de memória (RAS0-RAS3) e simultaneamente aceder à posição de memória determinada pelo endereço de linha e coluna Li e Ci. Note-se que, durante um acesso normal à célula Li,Ci (RAS0 e CAS0), todas as células da linha (Li) do Banco0, estarão também a ser refrescadas!.

No ciclo seguinte de Scrubbing Refresh, a célula a ser acedida, corresponderá à linha seguinte (Li+1, Ci) do Banco0, enquanto as linhas Li+1 dos demais bancos (RAS1-RAS3), estão a ser refrescadas.

Desta forma, não só o ciclo de Refresh (do tipo 'RAS-Only') é preservado (i.e. garante-se que todas as células de memória sejam refrescadas com uma periodicidade de 8ms), como também se possibilita um acesso 'normal' às mesmas, embora que com uma periodicidade bastante inferior, mas suficiente para reduzir a taxa residual de erros. É este acesso normal, que ocorre durante o período de refresh, que efectua o 'scrubbing' da respectiva célula acedida. Se durante este acesso, o EDAC sinalizar a ocorrência de erro, o controlador deverá alongar o ciclo de refresh em questão, permitindo a re-escrita da informação corrigida em memória. Assim, para além do normal refresh dos bancos de memória, o ciclo de 'Scrubbing' compreende ainda as seguintes operações:

a) Ler, simultaneamente, o conteúdo da célula (32-bits de dados) da DRAM e os 7-'*check bits*' da DRAM complementar;

b) Activar o EDAC para detectar e corrigir eventuais erros na palavra armazenada;

c) Dependendo do resultado 'No Error', 'Single-Error' ou 'Multiple-Error', gerado pelo EDAC, o controlador poderá decidir ou não, desencadear um ciclo de re-escrita dos dados para corrigir o erro. Esta correcção só será, no entanto, possível no caso de haver um 'erro simples'.

B. Funcionamento do EDAC

B.1 Ciclo de Escrita

Na operação de escrita para a memória a unidade de detecção e correcção de erros (EDAC) gera *check bits* a partir dos bits de informação. Para cada palavra de 32 bits a EDAC executa um ciclo por forma a gerar 7 *check bits* através de um algoritmo de Paridade de Hamming. Os *check bits* são o overhead envolvido na utilização de uma unidade de detecção e correcção de erros, mas o seu cálculo e armazenamento é necessário para a reconstrução da informação. A palavra armazenada em memória terá mais 7 bits, o que implica ocupação de espaço por informação extra, os bits de redundância.

B.2 Ciclo de Leitura

Ao realizar um ciclo de leitura os 39 bits, data+*check bits* são lidos da memória. Com os bits de data, novos *check bits* são gerados pelo circuito EDAC com a finalidade de testar a validade da informação. A validação é uma simples operação XOR que detecta a presença de bits diferentes e produz um *síndroma* que indica a posição de erro.

Check Word Bit	32 Bit Data Word																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB0	x		x	x		x					x		x	x	x		x			x			x	x	x	x		x				x
CB1				x		x		x		x		x		x	x	x					x		x		x	x		x	x	x		
CB2	x		x			x	x		x			x	x				x	x		x			x	x		x			x	x		x
CB3			x	x	x				x	x	x				x	x			x	x	x					x	x	x			x	x
CB4	x	x							x	x	x	x	x	x			x	x								x	x	x	x	x	x	
CB5	x	x	x	x	x	x	x	x									x	x	x	x	x	x	x	x								
CB6	x	x	x	x	x	x	x	x																	x	x	x	x	x	x	x	x

Figura 23 - Tabela do algoritmo de paridade de Hamming.

Esta tabela tem 32 colunas correspondentes aos 32 bits da palavra de informação. Debaixo de cada posição de bit 0 a 31, teremos o valor do *síndroma* que é gerado

quando houver erro numa destas posições (x corresponde a 1 na tabela e vazio ou espaço corresponde a 0).

B.3 Ciclo de Leitura-Correcção-Escrita

Este ciclo pode ser decomposto em três fases:

B.3.1 Read & Flag

Após ter dado início a um ciclo de leitura da DRAM, o controlador põe o EDAC a funcionar no modo de Read & Flag, colocando S0 a low e S1 a high.

Durante o ciclo de Read &Flag (leitura-assinala se há erro), o EDAC lê, simultaneamente, os bits de informação DQ0..DQ31 da DRAM e os respectivos check-bits CBQ0..CBQ6 da memória CBMEM. A partir dos bits de dados, através do Check Bit Generator, gera novos *check bits*. O Syndrome Generator, compara os novos check-bits, com os *check bits* lidos da memória, produzindo assim o *sindroma*.

Este é em seguida analisado no Error Detector. Caso exista erro num só bit de dados, esse acontecimento é assinalado através da Flag ERR. Se se detecta a ocorrência de um erro duplo, assinala-se através da Flag MERR.

Em função desta informação, o controlador da DRAM decidirá se deve proceder ou não à correcção da informação. Interessa aqui considerar a situação em que existe erro num só dos bits (Flag ERR).

B.3.2 Correcção

Esta operação é inicializada no mínimo 5ns após a entrada dos bits de informação e *check bits* (fase anterior), comutando S0 para high mantendo S1. Na fase de correcção, a informação em Q0..Q31 e CBQ0..CBQ6 ter-se-á de manter válida pelo menos 10ns.

O síndroma que já tinha sido calculado e usado para sinalizar a ocorrência ou não de erros, no Error Detector, é agora também introduzido no Bit-in-Error Decoder, que determinará qual dos 32 bits está errado.

A saída do Bit-in-Error Decoder é uma palavra de 32 bits com um bit a "1". Esta palavra sofre uma operação XOR com a palavra de informação original (razão pela qual se manteve válida nas linhas os 10ns após já termos entrado na fase de correcção), no bloco Error Corrector que não é mais do que 32 XORs em que um deles altera o bit em erro indicado pelo *síndroma*.

No final desta fase temos, em D0..D31 os 32-bits de dados corrigidos e em CBD0..CBD6 o *síndroma*.

B.3.3 Escrita

Antes que se possa armazenar a palavra corrigida na DRAM, é necessário ainda colocar nas linhas CBD0..CBD6 não o síndroma, mas sim os novos check-bits correspondentes à palavra corrigida. Para isso, é necessário colocar o EDAC em modo de escrita (ambos S1 e S0 a low).

Finalmente, escreve-se a informação e os respectivos check-bits corrigidos na DRAM, activando a linha de 'Write' (W).

Em resumo, enquanto a DRAM passa por um ciclo de 'Read-Modify-Write', o EDAC passa pelas fases de 'Read &Flag', Correcção e Escrita. Este ciclo está representado na Fig. 24.

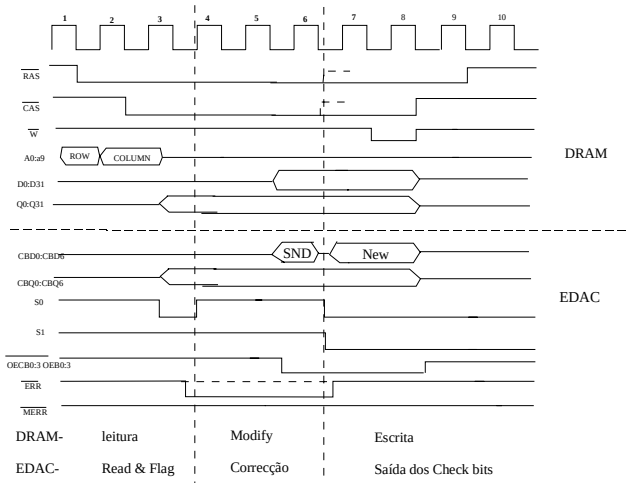

Figura 24 - Diagrama temporal da EDAC.

C. Diagrama de blocos do EDAC

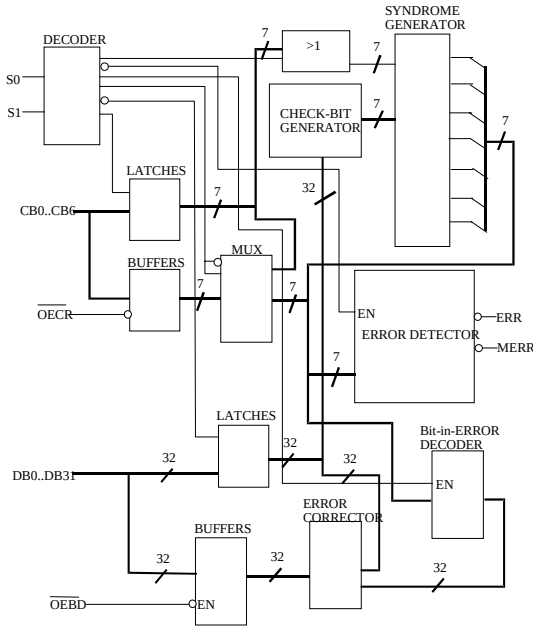

Figura 25 - Diagrama de blocos da EDAC.

No diagrama da EDAC os blocos fundamentais são:

- Decoder: descodifica os sinais de entrada S0 e S1.
- Check bit generator: gera 7 *check bits* a partir duma entrada de 32 bits.
- Syndrome generator: Faz um XOR dos bits do check bit generator com bits lidos de memória vindos de CB0..CB6. Na saída obtém-se o *síndroma* indicando a posição de erro na palavra.
- Error detector: detecta a ocorrência de erro simples ou múltiplo consoante o *síndroma* recebido do syndrome generator.
- Bit-in-error decoder: usa os *check bits* para gerar uma palavra de 32 bits que serve de "corrector", indicando a posição de erro na palavra.
- Error corrector: corrige a palavra de data, vinda de DB0..DB31, através de uma operação XOR com a palavra "correctora" do bit-in-error decoder.

D. Sinais do EDAC

Seleccionam o modo de operação da EDAC.	S0-S1
Check bits do código de erro.	CB0-CB6
Selecção de entrada e saída da informação.	OEB0-OEB3
Flag que sinaliza um único erro.	ERR
Flag que sinaliza erro múltiplo.	MERR

E. Alterações ao Controlador

Para incluir a possibilidade de gerar ciclos de 'Scrubbing Refresh' o Controlador e o respectivo Diagrama de estados deverão ser alterados para:

- incluir no contador que gera os endereços de refresh, a possibilidade de também gerar endereços de coluna, para além dos de linha, durante um ciclo de refresh;
- gerar o timing de RASi e CASi e Wi, necessário ao Scrubbing dos Bancos de memória;
- gerar os sinais de S0 e S1 que controlam o EDAC;
- actuar sobre os sinais provenientes do EDAC, por forma a alterar ou não os ciclos de leitura (os quais poderão exigir ciclos de re-escrita dos dados para correcção).

Uma alteração possível para o diagrama de estados do controlador, está indicada na Fig. 26.

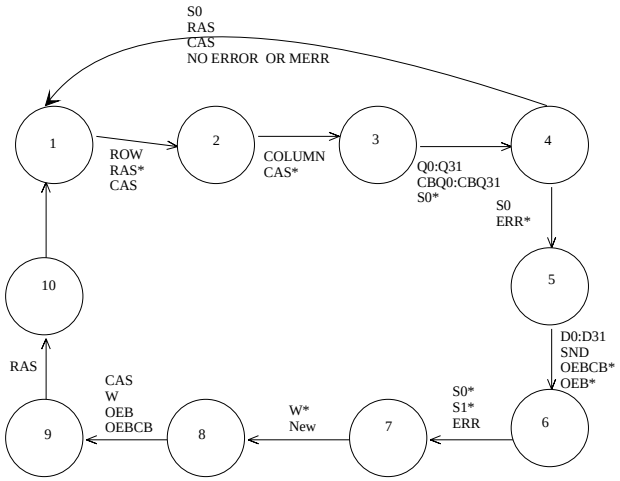

Figura 26 - Diagrama de estados do ciclo de scrubbing refresh.

VII. CONCLUSÕES

Como se poderá avaliar através exposição que acabámos de fazer, o design dum controlador de DRAM envolve um conjunto de conceitos, por si só que merecem uma abordagem mais aprofundada deste tipo de assunto.

Este artigo pretende servir de introdução, a alguma da problemática que envolve o design com as modernas RAMs dinâmicas, nomeadamente no tocante aos modos de acesso, 'Nibble Mode', 'Page Mode', etc, bem como aos tipos de 'refresh' possíveis, 'Ras-Only' e 'Scrubbing Refresh'. Este último, suscita por outro lado a compreensão do timing e algoritmos (Hamming) necessários à detecção e correcção de erros. O 'overhead' envolvido para aumentar a confiabilidade das DRAMs é hoje em dia perfeitamente justificado, uma vez que a probabilidade de ocorrência de 'soft-errors' (nomeadamente devida à emissão de partículas alfa) aumenta bastante com a densidade de células por chip de memória.

O principal desafio deste projecto, passou pela compreensão do funcionamento de alguns tipos de DRAM disponíveis, entendimento dos protocolos de interface ao bus dum microprocessador do tipo do MC68030, para poder gerar uma especificação para o

controlador. Para testar a validade desta especificação, o mesmo foi simulado com a ajuda da linguagem de descrição de hardware ABEL da Data I/O.

REFERÊNCIAS

[1] Data sheets, Dynamic RAM's, Texas Instruments, 1986.
[2] Technical Summary, Second Generation 32-bit Enhanced Microprocessor, Motorola Semiconductor, 1986.
[3] Notas da cadeira de opção: Técnicas de Interface, 1995.
[4] EDN, Designer's Guide - Dynamic RAMs, Parte I a IV, 1989.