Placa Quadriprocessador Baseada em Transputers

José C. Fonseca , Paulo J. Gonçalves
António R. Borges , Artur C. Pereira , Joaquim A. Martins , Joaquim S. Pinto

Resumo **- Neste artigo, é descrito um módulo multiprocessador baseado em transputers. Tem quatro processadores, cada qual com memória local, memória partilhada e interfaces a *PC-AT* e *Ethernet*. A proposta principal é a de construir um bloco (nó) com uma arquitectura *message-passing* de modo a evitar a interligação em malha *2-D*. Foram tomados cuidados especiais para minimizar o número de chips envolvidos, bem como usadas as características especiais apresentadas pelo transputer no que diz respeito às comunicações interprocessador e concentrada toda a lógica de controlo num *ASIC* multifuncional. A especificação da placa, a análise racional do seu desenho e *pin-out* do *ASIC* estão apresentados e descritos em grande detalhe.**

Abstract **- In this paper, a transputer-based multiprocessing module is described. It has four processors, each with local memory, a shared memory and interfaces for PC-AT and Ethernet. The main purpose is to use it as a building block (node) for a message-passing architecture saving a 2-D mesh interconnection. Special care was taken to minimize chip count, both by using the special features presented by the transputer family for interprocessor communication and by concentrating all controlling logic in a multipurpose ASIC. The board specification, the rationale of its design and the ASIC pin-out are presented and described in full detail.**

I. INTRODUÇÃO*

Hoje em dia são cada vez mais comuns as aplicações para computadores e, ao mesmo tempo, mais ambiciosas devido aos progressos da ciência da computação, necessitando que se desenvolvam computadores cada vez mais poderosos. Algumas das aplicações que estão em franca expansão são as denominadas aplicações computacionais intensivas como sejam o processamento de imagem, síntese de imagens, realidade virtual, imagem médica, visão por computador, resolução de modelos dinâmicos, física da atmosfera, aerodinamismo, astronomia, controlo industrial, etc., nas quais a intensidade de processamento pode atingir valores muito elevados. Esta capacidade de processamento só pode ser conseguida usando processadores tecnologicamente muito avançados ou então arquitecturas que recorram ao processamento paralelo usando vários processadores, como se mostra na figura 1. Desta última forma obtêm-se resultados semelhantes aos da a primeira usando-se componentes mais standardizados tornando-se muito mais económica.

Para cobrir especialmente esta área de arquitectura, em 1985 o fabricante de *LSI InMos* (Reino Unido) colocou no mercado um *CPU* cuja arquitectura se pode considerar inovadora, à qual deu o nome *transputer*. O nome escolhido pretende realçar as suas características principais: *TRANSport* e *comPUTER*. O *transputer* é um circuito integrado *VLSI* com processador, memória e *links* de comunicação para ligações directas com outros *transputers* e preparado para multiprocessamento.

O presente trabalho descreve o projecto de uma placa multiprocessador baseada em *transputers*, mais precisamente em quatro *IMS T800* da *InMos* [1], placa essa que possui também *interfaces* para *bus PC* [2] e rede *ethernet*. Esta placa está dotada de um sistema de memória partilhada, pois o processo de transferência de dados entre processadores via *links* não é muito eficiente no que diz respeito às taxas de transferência.

No desenvolvimento de uma placa deste tipo deve-se ter em conta as suas dimensões e viabilidade económica. Desta forma e, de modo a reduzir o espaço, o projecto teve como base o desenho de *Aplication Specific Integrated Circuits* (*ASIC*) que englobam tanto quanto possível toda a lógica necessária. Como ferramenta de desenho assistido por computador foi usado o *Cadence* da *European Silicon Structures (ES2)* [3]. O *ASIC* foi desenhado em *semicustom* pois permite projectar sistemas de grande complexidade e dimensão com um tempo de projecto bastante menor do que o *fullcustom*.

A placa poderá vir a constituir um módulo de processamento que terá a capacidade de ligação a outras

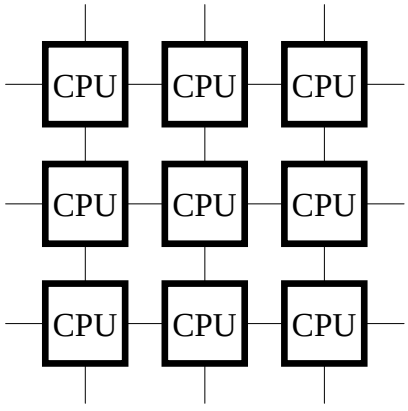


FIGURA 1
TOPOLOGIA CONVENCIONAL DUMA MALHA *2-D* DE PROCESSADORES.

* Trabalho efectuado no âmbito da disciplina de projecto

placas semelhantes pela utilização dos quatro *links* que estarão disponíveis para o exterior. Assim sendo será possível criar uma rede bidimensional de placas que desta forma terão uma capacidade muito superior à de um só elemento.

II. OBJECTIVOS.

Tal como o nome do projecto sugere, "*Placa quadriprocessador baseada em transputers*" ou simplesmente "*Macrotransputer*", o que se pretendeu desenvolver foi uma placa protótipo constituída internamente por quatro processadores do tipo *transputer*, memória e lógica de *interface*. É natural que se pretendam aproveitar as facilidades e potencialidades dos *transputers*, nomeadamente a facilidade de comunicações entre si e a elevada velocidade de processamento. A figura 2 representa a topologia *standard* para uma situação onde se encontram quatro *transputers* interligados, formando um bloco mais complexo e potente.

Como se pode ver a ligação é feita muito facilmente através dos canais série (*links*) que são ligados directamente de um *transputer* a outro, sem necessidade de lógica adicional, ou seja, para ligar um *transputer* a outro só é necessário ter os dois *transputers*.

A placa que se descreve tem funcionalmente os quatro *transputers* ligados da forma acima descrita, mas é diferente desta num aspecto essencial que é a inclusão de uma memória partilhada por todos os processadores em vez de só possuírem memória local a cada um. Esta diferença pode parecer, à primeira vista, pouco significativa, mas rapidamente se torna evidente a grande vantagem desta topologia que é a de tornar acessível, virtualmente ao mesmo tempo e aos quatro processadores, um mesmo bloco de dados aos quais os *transputers* vão poder aceder em paralelo, permitindo uma distribuição de tarefas entre eles e uma mais rápida conclusão das mesmas.

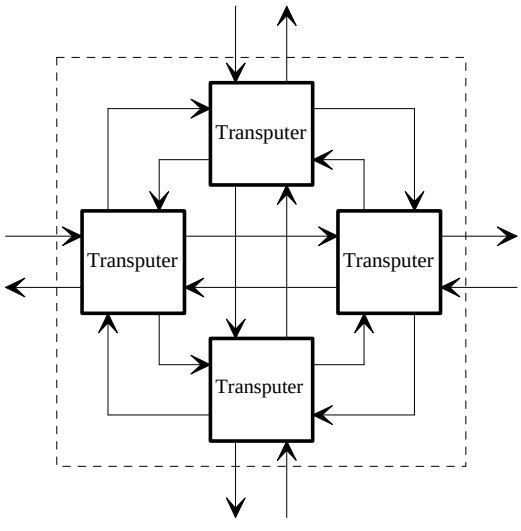


FIGURA 2

NÓ DE QUATRO *TRANSPUTERS* LIGADOS ATRAVÉS DOS SEUS CANAIS SÉRIE

Durante a execução das tarefas, os *transputers*, poderão comunicar entre si para se sincronizarem, trocarem informações e resultados intermédios, ou ainda executarem outras tarefas. Esta topologia, no entanto, tráz também consigo algumas desvantagens e limitações, pois, actualmente, as memórias que permitem acessos *multi-port*, têm capacidades de armazenamento muito reduzidas o que as torna impróprias para este caso. A solução para esta situação foi a de simular estas memórias com base em memórias convencionais e um controlador de acesso (figura 3). Neste caso torna-se imprescindível a existência de um mecanismo que coordene quem vai ser atendido num determinado instante, quem tem de esperar a sua vez, quem se segue na fila de espera, enfim, coordenar todas os problemas e decisões que possam aparecer. Designámos esse controlador de acesso por árbitro e, como é de prever, ele é uma das peças chaves do sistema e da qual depende o seu melhor ou pior desempenho.

Tendo em vista aumentar ainda mais a área de trabalho onde poderá ser usado uma placa deste tipo, a sua funcionalidade e tipo de utilizadores, aparecem ainda mais duas entidades que vão funcionar em paralelo com os *transputers*. A primeira é um *interface* para *bus PC*, de modo que, a partir de uma ferramenta tão divulgada como seja um *PC*, seja possível comunicar e transferir dados de informação ou controlo com a placa. A segunda, não sendo menos importante, é um *interface* para rede *ethernet* aumentando ainda mais a gama de acção que se conseguiria com os *links* e alargando a utilidade da placa.

O que pretendemos desta placa é que ela permita a transferência, manipulação, tratamento e processamento de grandes quantidades de informação de uma forma rápida e sincronizada. Além disso possibilita-se a recepção, transmissão e controlo de dados de diversos periféricos tais como o *bus PC*, a rede *ethernet* e outros *transputers*.

Um ponto interessante de salientar é o facto de que, tal como no caso dos *transputers*, deve ser possível ligar várias placas entre si ou a *transputers* formando uma rede com *performances* muito superiores à de um único elemento. Poder-se-á substituir e com vantagem, por exemplo, redes que tenham sido projectadas para *transputers* por redes baseadas nesta placa multiprocessador.

Pretendia-se que a lógica de controlo deste sistema fosse implementada sob a forma de um *ASIC* de forma que, tanto quanto possível, se evitassem componentes discretos. Este desenvolvimento deveria ser feito com base em captura de esquemáticos usando células de biblioteca (*standard cells* e *macro cells*). Como ferramenta de trabalho para captura destes esquemáticos e respectivas simulações usa-se o pacote de *software* de *CAD* da *Eurochip*, o *Cadence*, para a configuração da *ES2* e tecnologia de 1.0μ [4]. Trata-se de uma ferramenta que é geral no que diz respeito à tecnologia que usa e aos fabricantes e está em fase de divulgação e expansão no nosso país.

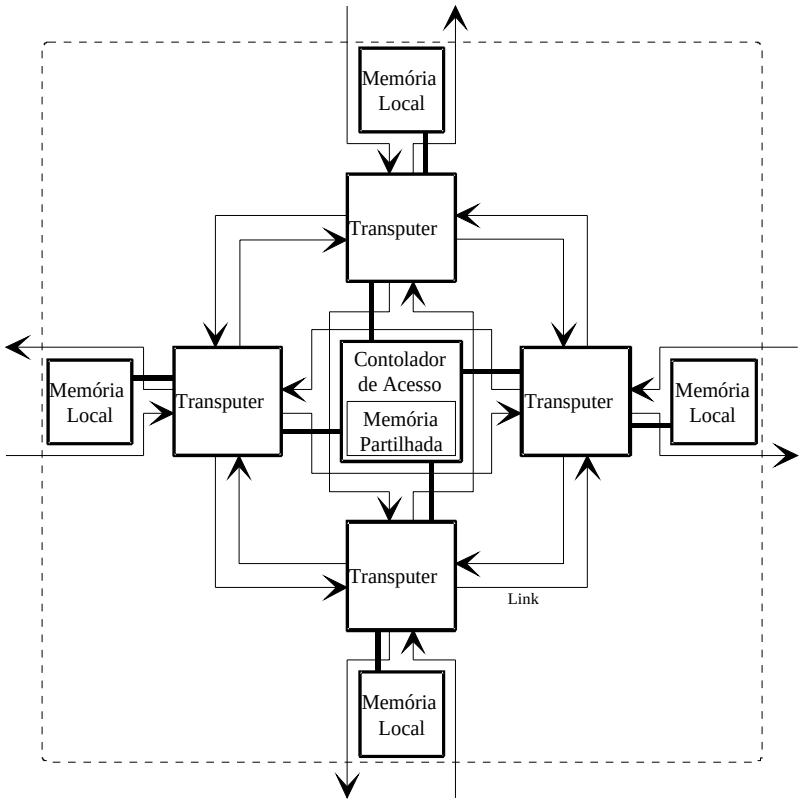


FIGURA 3
REPRESENTAÇÃO DAS ÁREAS DE MEMÓRIA, ÁRBITRO E *LINKS*.

III. ESPECIFICAÇÃO DO SISTEMA

A. Especificação funcional

À partida o sistema é constituído por quatro *transputers*, cada um deles com um *interface* para memória dinâmica local de 4Mbyte ou 16Mbyte e um outro *interface* para um bloco de memória dinâmica partilhada também de 4Mbyte ou 16 Mbyte [5]. O acesso a esta é controlado por um elemento designado por árbitro, tratando com igual prioridade os vários pedidos de acesso seguindo um mecanismo *round-robin*. Estas memórias necessitam de ciclos de *refresh*, pelo que no caso da memória local, é o próprio *transputer* o reponsável por estes, mas no caso da memória partilhada existe uma entidade específica para esse fim. O acesso desta entidade à memória dinâmica partilhada para execussão de ciclos de *refresh* é feito em concorrência com os restantes utilizadores desta memória, pelo que necessita também de ser controlado pelo árbitro. O árbitro, devido à natureza da entidade em questão, concede a permissão de acesso da forma mais rápida, tratando o pedido desta com prioridade sobre os restantes pedidos de acesso.

Este sistema está ainda dotado de um *interface* para *bus PC*, possibilitando o acesso transparente, sob o ponto de vista do utilizador, à memória dinâmica partilhada. Este acesso é feito por endereçamento directo do CPU do PC em questão a uma área de memória de 32Kbyte que corresponde a uma página da memória partilhada. Essa página é seleccionavel através de registos I/O do PC de forma que é possível aceder a qualquer posição da memória partilhada. Os acessos são feitos sempre a oito bits.

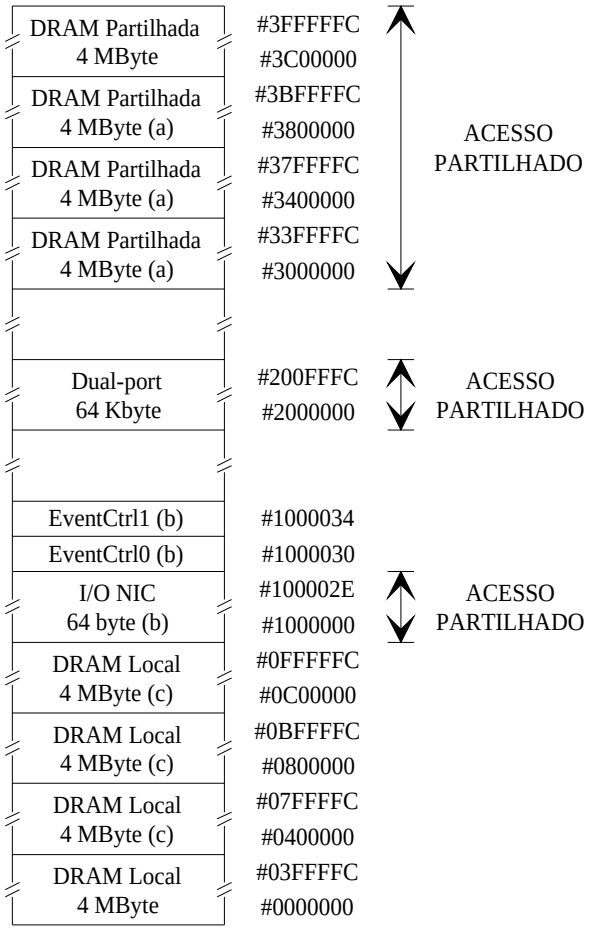
O *interface* para rede *ethernet* é feito através de um controlador de rede (*NIC - Network Interface Controller*). Neste caso em concreto, é usado o DP8390C da *National Semiconductor* [6]. Este interface pode ser visto como duas entidades sendo a primeira tomada como passiva. Esta entidade trata-se de um conjunto interno de registos para programação e *status* do *NIC*. Para maior flexibilidade do sistema, estes registos são vistos da mesma forma que a memória partilhada, sendo o acesso a estes possível de ser concretizado por qualquer *transputer* ou mesmo através do *bus PC* e de igual forma controlado pelo árbitro. A segunda entidade é activa e está ligada ao restante sistema através de uma memória *dual-port* [7]. Por parte do *NIC*, esta memória é usada como o seu buffer externo, possibilitando o armazenamento de pacotes. Por parte do restante sistema é vista como mais uma área de memória partilhada.

Os dois *interfaces* adicionais, tanto para *bus PC* como para rede *ethernet* através do *NIC*, vieram impor a existência de um processo de comunicação assíncrono entre estes e os *transputers*, já que os *transputers* têm entre si os *links* para esse efeito. Assim, no caso do *transputer* é usado o sinal de *Event* e no caso do *bus PC* um dos sinais *IRQ*. O sinal de *Event* pode ser activado pelo controlador de rede ou pelo *PC* e o sinal *IRQ* pode ser também activado por parte do controlador de rede ou ainda por qualquer um dos quatro *transputers*. Como os sinais de entrada para reconhecimento de interrupções são únicos, para serem reconhecidas individualmente cada uma das fontes, existe um registo (*EventCtrl0*), que através da sua leitura posterior se poderá determinar qual a origem da fonte de interrupção. O sinal de interrupção por parte do controlador de rede é gerado por meio de *hardware* através de um pino que este possui. No caso de sinais de interrupção entre *PC* e *transputers* (e *transputers* e *PC*), estes são gerados pela escrita num registo local (*EventCtrl0*) de uma palavra que deve conter o destino da mesma. Esse registo é individual, não sendo necessário portanto recorrer ao controlador de acesso para o seu uso. Está ainda implementado um outro registo (*EventCtrl1*) para tornar possível mascarar fontes de interrupção não desejadas.

Para aceder aos vários dispositivos existentes na placa, cada *Transputer* pode endereçar ao mapa de memória que se encontra indicado na figura 4. No caso do *bus PC*, a área da memória local não tem significado, no entanto os restantes dispositivos encontram-se distribuidos da mesma forma. No caso dos *transputers*, o mapeamento é directo, correspondendo a posição #0000000 ao endereço #00000000 do *transputer*. Dado se desprezarem na descodificação os bits de endereço de 26 a 31, a memória dinâmica local pode ser vista no endereço #80000000 que é a base do *transputer*, assim como a memória dinâmica partilhada pode ser vista em #7FC00000 ou #7F000000 caso se trate respectivamente de 4Mbyte ou 16Mbyte indo assim até ao topo de endereçamento deste. Convém referir que quando a memória local é acedida a partir do endereço #80000000, o *transputer* não reconhece a área compreendida entre #80000000 e #80000FFF pois trata-se do espaço de endereçamento à sua memória interna e que é usada entre outras coisas para guardar os registos do processador. Para fazer uso da memória dinâmica local na totalidade, basta aceder por exemplo através do endereço #00000000. O *PC* faz o acesso através de uma área de 32Kbyte, que é reconhecida como memória local do mesmo, nos endereços de #000000 a #007FFF sendo os nove bits de maior peso configuráveis de forma que esta área não entre em conflito com zonas já ocupadas. Esta área equivale a uma página da área ocupada pelos dispositivos existentes na placa podendo ser definida através de um registo.

A nível funcional, o diagrama de blocos que se propõe é o indicado na figura 5. A memória dinâmica local está organizada em quatro bancos de 1Mbyte ou 4Mbyte, o que perfaz 4Mbyte ou 16Mbyte respectivamente. É a memória que cada transputer tem como sendo particular e o seu acesso é feito sem necessidade de inserção de ciclos de espera nem arbitragem. O *interface* para memória local contém a lógica necessária para a ligação da memória dinâmica local ao *transputer*. Contém uma unidade de registo e multiplexagem de endereços capaz de gerar linhas e colunas. De todos os restantes sinais destinados ao controlo da *DRAM*, apenas o sinal de *CAS* é tratado, com o fim de selecção da referida memória entre os restantes dispositivos. O *interface* para acesso partilhado possui duas funções. Uma delas é a de fazer o tratamento referente a pedidos de *Event* ou *IRQ*. Para isso contém registos que podem ser acedidos por parte do *transputer* ou *PC* a este *interface* associado, tanto para leitura como escrita. Estes registos contêm vários *bits*, estando cada um deles associado a um significado, caso se trate de uma escrita ou leitura respectivamente. No caso de um dos registos (*EventCtrl0*) o processo de escrita possibilita gerar remotamente um pedido de interrupção, enquanto a leitura é usada para verificar o *status* das interrupções chegadas. O outro registo (*EventCtrl1*) serve para mascarar individualmente cada entrada de pedido de interrupção.

A segunda função é a de servir de porto para acesso de escrita e leitura nos recursos partilhados, que são a memória dinâmica partilhada, memória dual-port partilhada e espaço de endereçamento *I/O* do *NIC*. Para isso existe lógica que permite guardar e gerar endereços de forma simples ou multiplexada (endereço normal ou por linhas e colunas) e gerar sinais de controlo de uma forma temporal e autónoma, de forma a tornar possível a compatibilidade dos vários tipos de dispositivos partilhados com o *transputer* ou o *bus PC*, sendo esta última tarefa realizada com ajuda de máquinas de estados. Estas máquinas existem principalmente pelo facto de ser necessário garantir uma determinada evolução temporal dos sinais de controlo quando o acesso é desencadeado. Incluído no porto para escrita e leitura, está o sistema que gera o pedido de acesso. O pedido é activado quando se detecta que o dispositivo (*transputer* ou *PC*) pretende realizar um acesso a um endereço referente a um recurso partilhado. Este pedido de acesso destina-se ao



a) Só para memória dinâmica partilhada de 16Mbyte.
b) Apenas existem os 8 bits menos significativos.
c) Só para memória dinâmica local de 16Mbyte.

FIGURA 4
MAPA DE MEMÓRIA.

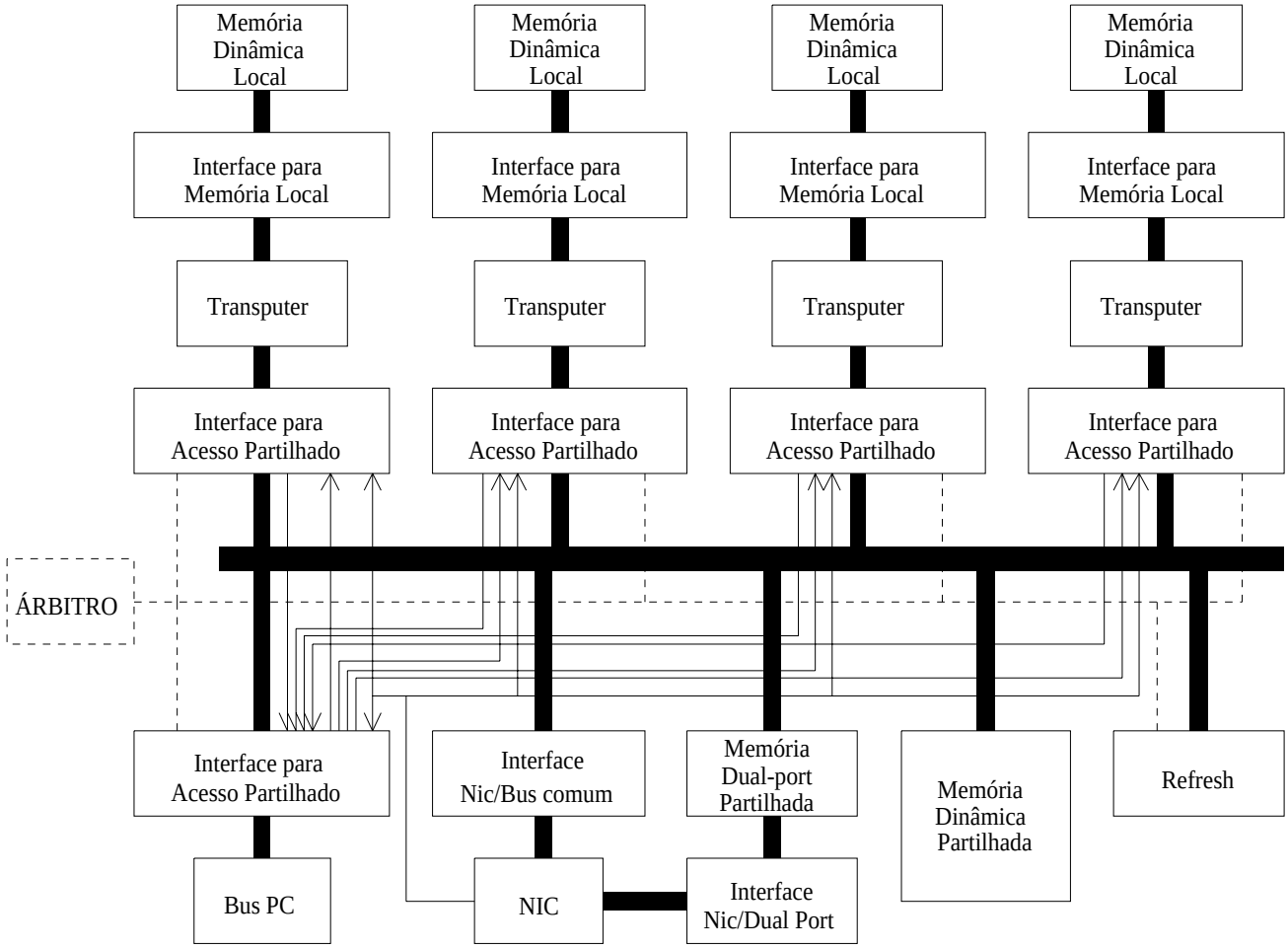


FIGURA 5
DIAGRAMA DE BLOCOS SEM OS *LINKS*.

controlador, designado por árbitro, que por sua vez, enviará um sinal de confirmação no momento que for possível esse mesmo acesso. Este sistema está equipado com a capacidade de tratamento a nível de *time-out*, de forma que a não atribuição de confirmação de acesso a um recurso partilhado não provoca a falha do mesmo dispositivo que realizou o pedido. A falha neste caso está no facto dos *transputers* não poderem estar indeterminadamente em estado de *wait*, pois são responsáveis pelo refrescamento das suas memórias locais.O *Interface NIC* / *bus* comum permite isolar o *bus* do *NIC* do *bus* comum aos recursos partilhados. Isto deve-se ao facto do *bus* que o *NIC* usa para funcionamento autónomo através dos seus *DMA´s* para transferências com a sua memória externa (*dual-port*) ser o mesmo por onde se pode aceder aos registos internos deste controlador. Desta forma, quando o *NIC* se encontra em funcionamento autónomo, não existe necessidade de recorrer à arbitragem para uso do *bus* comum.

A memória dual-port partilhada trata-se de um bloco de memória de 64Kbyte do tipo *dual-port*, implementado com base em quatro bancos de 16Kbyte. Esta memória está organizada a 32 *bits*, embora por parte do *NIC* o acesso se efectue apenas a 8 *bits*. O uso de quatro bancos de 8 *bits* em vez de uma memória de 32 *bits*, é devido à inflexibilidade da segunda opção no que diz respeito a ciclos de escrita nos bancos individualmente.

O *interface NIC / dual-port* serve apenas para realizar o *latch* dos endereços do *NIC* e descodificar o banco ao qual se destina o mesmo, quando este pretende efectuar um acesso ao seu *buffer* externo, ou seja, à memória *dual-port*. Os acessos do *NIC* devem ser feitos sempre a 8 *bits*, embora este tenha potencialidade para os fazer a 16 *bits*, não devendo ser esquecido este facto na inicialização do mesmo, sob pena de incorrecto funcionamento por perda de dados.

O *Refresh* é responsável por gerar os pedidos de acesso à memória dinâmica partilhada para execução de um ciclo de refresh sobre a mesma, bem como a própria execução. O primeiro ponto é executado por uma unidade de temporização que gera periodicamente um pedido de acesso. A execução fica a cargo de uma máquina de estados que realiza a correcta sequência temporal equivalente a um ciclo *CAS Before RAS Auto Refresh*.

Este tipo de acesso dispensa uma unidade de contagem para indicação da linha de refrescamento.

O árbitro é responsável pela atribuição ordenada de permissões de acesso aos recursos partilhados. Nesta unidade todos os pedidos têm igual prioridade à excepção do pedido proveniente da unidade de *refresh* que é o mais prioritário. A atribuição nos restantes casos é tratada segundo uma ordem tipo *round-robin*.

B. Opções tomadas na implementação.

Sendo o projecto constituído por vários blocos bem definidos e com fins bastante específicos, inicialmente estaria prevista uma certa divisão das várias entidades que iriam constituir a placa em circuitos separados, ou seja, pretendia-se que os diferentes blocos constituíssem entidades fisicamente diferentes, muito embora comunicantes entre si. Com esta solução pretendia-se integrar as partes de arbitragem, de *interface* com o *transputer*, *bus PC* e ainda à rede *ethernet*. O *interface* para a memória local dos *transputers* seria feito com lógica discreta. Esta solução implicaria, possivelmente, a construção de quatro *ASIC's* diferentes o que seria inviável economicamente, pois ficaria demasiado dispendioso. Procuraram-se então soluções alternativas. A solução seguinte foi a de construir um único *ASIC* que englobaria a maior e mais complexa parte da lógica e dos

TABELA I
MODOS FUNCIONAIS DO ASIC.

Modo		
ArbE	**Mode**	**Blocos funcionais activos**
0	1	***Interface* para *transputer*.** • *Interface* para memória local. • *Interface* para acesso partilhado.
1	1	***Interface* para *transputer*.** • *Interface* para memória local. • *Interface* para acesso partilhado. • Árbitro. • *Refresh*.
0	2	***Interface* para *bus PC*.** • *Interface* para acesso partilhado.
1	2	***Interface* para *bus PC*.** • *Interface* para acesso partilhado. • Árbitro. • *Refresh*.
0	3	***Interface* para *NIC*.** • *Interface Nic / bus* comum. • *Interface Nic / dual-port*.
1	3	***Interface* para *NIC*.** • *Interface Nic / bus* comum. • *Interface Nic / dual-port*. • Árbitro. • *Refresh*.

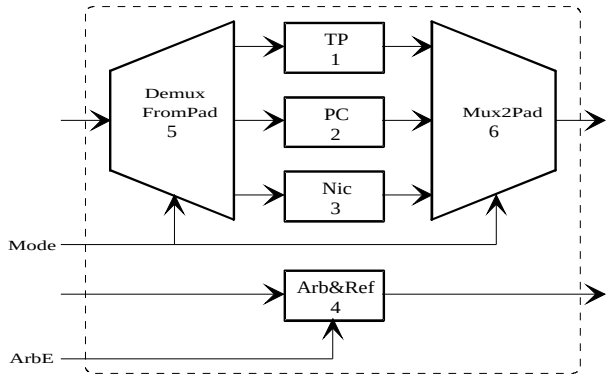


FIGURA 6
DIAGRAMA DE BLOCOS INTERNO DO *ASIC*.

circuitos necessários. Para complementar esse circuito integrado desenvolver-se-iam outros circuitos em lógica discreta que preenchiriam as necessidades de cada *interface* em particular. Esta solução parecia, à primeira vista, que resolveria o problema, mas tal não se verificou pois o número de pinos que o circuito deveria ter, superaria o número máximo que pode ser usado actualmente. Isto deve-se principalmente ao elevado número de linhas de dados e endereços. Por fim surgiu a ideia de desenhar um único *ASIC*, mas com uma filosofia radicalmente diferente desta última. A ideia base desta atractiva solução foi a de construir um *ASIC* que estivesse internamente dividido em várias unidades, cada uma tendo por objectivo realizar uma função específica. Para isso existem no *ASIC* alguns pinos de configuração que permitem definir um modo de funcionamento entre os vários possíveis. Esta opção torna-se especialmente atractiva, pois é possível reduzir bastante o custo do projecto, já que este implica a construção de apenas um *ASIC*. Em termos gerais, o *ASIC* está implementado da forma que a figura 6 ilustra.

De acordo com o modo pretendido é possível efectuar um dos *interfaces* e ainda activar ou não a parte de arbitragem e a unidade de refresh da memória dinâmica partilhada. A tabela I ilustra os vários blocos funcionais que são permitidos obter em cada modo. Pelo facto do ASIC ter vários modos de funcionamento, muitos dos pinos deste tomam funções distintas como se pode ver na tabela II. Notar que os modos de teste apresentados nesta tabela foram implementados com o fim de mais facilmente detectar e isolar erros do *ASIC*.

A placa quadriprocessador tem vários *ASIC's* destes, um para cada *interface* necessário, funcionando de acordo com as respectivas configurações e onde, num e apenas num só, a parte de arbitragem e refrescamento da memória dinâmica partilhada deve estar activa, não interessando no entanto em qual deles. O diagrama de blocos da figura 7 ilustra como na realidade é construida a placa quadriprocessador.

Na introdução do *interface* para *bus PC*, foi de início adoptada a ideia deste ser visto como um sexto dispositivo a disputar a memória dinâmica partilhada. A opção de

TABELA II
DISTRIBUIÇÃO DOS PINOS DO *ASIC*.

Transputer					PC					Nic				
Normal	Teste	I	O	T	Normal	Teste	I	O	T	Normal	Teste	I	O	T
Addr<0:13>	Addr<0:13>		X	X	Addr<0:13>	Addr<0:13>		X	X	Ad32<0:13>	Ad32<0:13>		X	
Data<0:7>	Data<0:7>	X	X	X	Data<0:7>	Data<0:7>	X	X	X	Data<0:7>	Data<0:7>	X	X	X
Data<8:31>	Data<8:31>	X	X	X	Data<8:31>	Data<8:31>	X	X	X					
notNicE	notNicE		X	X	notNicE	notNicE		X	X					
notMemDPE	notMemDPE		X	X	notMemDPE	notMemDPE		X	X					
notRd	notRd		X	X	notRd	notRd		X	X					
notWr<0:3>	notWr<0:3>		X	X	notWr<0:3>	notWr<0:3>		X	X	notCEB<0:3>	notCEB<0:3>		X	
notRas	notRas		X	X	notRas	notRas		X	X	notCas	notCas		X	
notCas	notCas		X	X	notCas	notCas		X	X	notRas	notRas		X	
notResReq	ColnotRow		X		notResReq	ColnotRow		X			Bst1Out		X	
notResAck	notResAck	X			notResAck	notResAck	X				BstTest	X		
BusyNic	BusyNic	X			BusyNic	BusyNic	X			notNicE	notNicE	X		
Reset	Reset	X			Reset	Reset	X			Reset	Reset	X		
MClock	MClock	X			MClock	MClock	X			MClock	MClock	X		
MemAd<0:7>	MemAd<0:7>	X	X		SD<0:7>	SD<0:7>	X	X		Ad<0:7>	Ad<0:7>	X	X	
MemAd<8:15>	MemAd<8:15>	X	X		SA<0:7>	SA<0:7>	X	X		Ad<8:15>	Ad<8:15>	X	X	
MemAd<16:31>	MemAd<16:31>	X	X		SA<8:23>	SA<8:23>	X	X						
notMemRd	notMemRd	X			notMemSR	notMemSR	X			notRd	notRd	X		
notMemWrB0	notMemWrB0	X			notMemSW	notMemSW	X			notWr	notWr	X		
notMemWrB<1:3>	notMemWrB<1:3>	X			IntTp<1:3>	IntTp<1:3>	X							
notMemS0	TestMemSE	X			AEN	TestMemSE	X			ADS0	ADS0	X		
notMemS1	TestMemDPE	X			IOCfg0	TestMemDPE	X				BstClk	X		
notMemS2	TestNicE	X			IOCfg1	TestNicE	X			notAck	notAck	X		
notMemS3	notTestFinish	X			IOCfg2	notTestFinish	X				notTestReqAkRf	X		
notMemS4	notMemS4	X			IOCfg3	IOCfg3	X				notTestRfReq	X		
notMemWait	RWE		X		notIOChRdy	RWE		X			Bst2Out		X	
MemS	MemS	X			MemS	MemS	X							
MemL	MemL	X			notIOW	notIOW	X							
notCasL	End		X		EventTp1	End		X			notRfReq		X	
AddrL<0:8>	AddrL<0:8>		X		MemCfg<0:8>	MemCfg<0:8>	X							
AddrL<9:10>	AddrL<9:10>		X		EventTp<2:3>	EventTp<2:3>		X						
EventNic	EventNic	X			IntNic	IntNic	X							
EventPc	EventPc	X			IntTp0	IntTp0	X							
Event	ContTp		X		Int	ContPc		X			ReqAkRf		X	
IntPc	Idle		X		EventTp0	Idle		X						
					ALE	ALE	X							
Mode<0:1>	Mode<0:1>	X			Mode<0:1>	Mode<0:1>	X			Mode<0:1>	Mode<0:1>	X		
Test	Test	X			Test	Test	X							
notArbE	notArbE	X			notArbE	notArbE	X			notArbE	notArbE	X		
notRemAck<0:4>	notRemAck<0:4>		X		notRemAck<0:4>	notRemAck<0:4>		X		notRemAck<0:4>	notRemAck<0:4>		X	
notRemReq<0:4>	notRemReq<0:4>	X			notRemReq<0:4>	notRemReq<0:4>	X			notRemReq<0:4>	notRemReq<0:4>	X		

I - ENTRADA ; O - SAÍDA ; T - TRISTATE

realizar o *interface* através de um dos *links* de um dos *transputers* pertencente ao sistema através de um *link adaptor* estaria fora de questão, devido à ocupação relativa de um link necessário para outros fins e a troca de informação não ser suficientemente rápida.

Convém referir que, dos quatro *links* que cada *transputer* possui, três são utilizados para comunicação assíncrona entre os *transputers* do sistema. Os quatro *links* que ficam disponíveis servem para que, em aplicações futuras, se possa proceder à ligação deste sistema em malha com outros sistemas idênticos. Sob o ponto de vista do *PC*, este vê a memória partilhada como sendo a sua. Como a memória dinâmica partilhada pode ter uma dimensão até 16Mbyte, nunca seria possível mapea-la no espaço de endereçamento de um *PC*, até porque parte deste já está ocupado. Desta forma, o acesso é feito por paginação de 32Kbyte. Para selecção da página que se deseje aceder são usados registos mapeados no espaço de endereçamento *I/O* do *PC*. O acesso será sempre a oito bits para evitar incompatibilidades existentes entre barramentos de vários

fabricantes de *PC* compatíveis que não respeitam rigorosamente o *standard*.

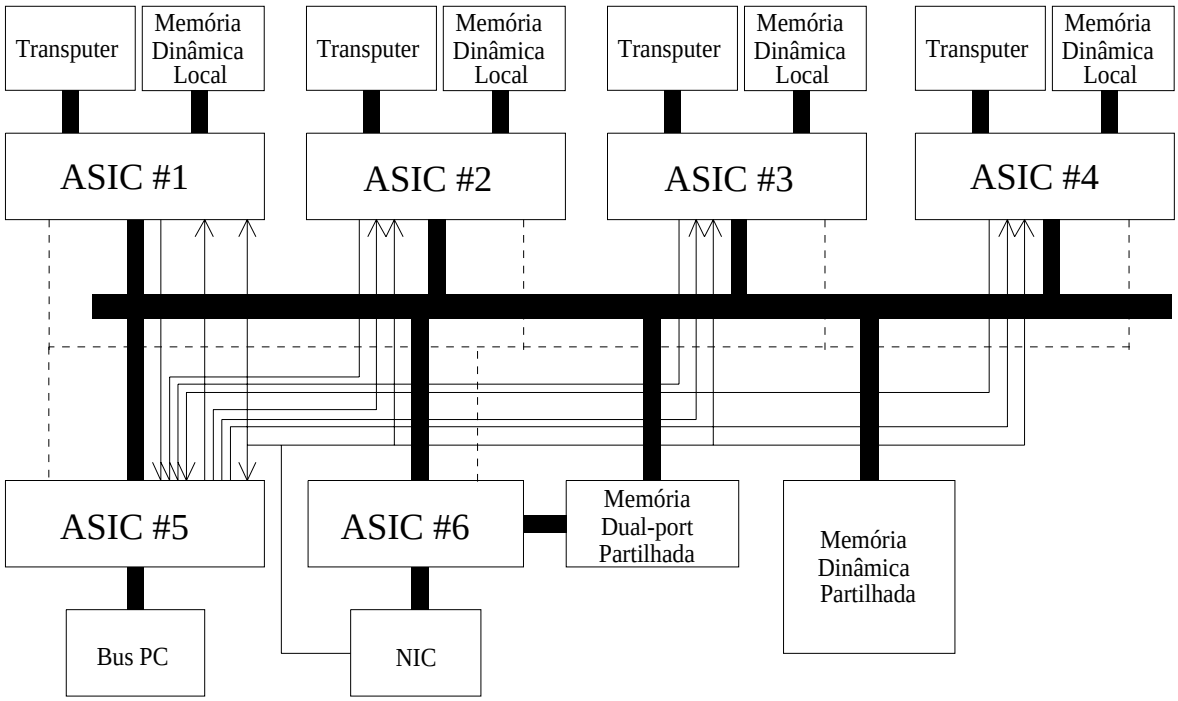


FIGURA 7
PLACA QUADRIPROCESSADOR BASEADA EM *TRANSPUTERS*.

Na especificação do *interface* para rede *ethernet* foi pensado inicialmente mapear a memória que o NIC necessitava para buffer externo na própria memória dinâmica partilhada, o que implicaria que o NIC teria de concorrer com os restantes transputers, ou mesmo com o bus PC, quando efectuasse um acesso ao seu buffer. Esta solução teve de ser abandonada por duas razões. A primeira tem a ver com os tempos de acesso à memória dinâmica partilhada que deveriam ser bastante curtos, pois o *NIC* possui um *buffer* interno muito pequeno.

A única solução seria tornar este utilizador da memória dinâmica partilhada também prioritário sobre os outros utilizadores. A segunda razão afecta cada um dos *transputers* e tem a ver com o tempo de ocupação pelo *NIC* da memória dinâmica partilhada em conjunto com os outros utilizadores, que poderá exceder o tempo máximo que um *transputer* pode ser mantido em lista de espera. Este tempo tem a ver com a necessidade que este tem de efectuar acessos periódicos à memória local para efectuar ciclos de refresh, já que esta também é dinâmica e ser o próprio *transputer* o responsável pela manutenção da mesma. A alternativa tomada para solucionar o primeiro problema relacionado com o tempo de acesso do *NIC* à memória dinâmica partilhada viria agravar o acesso por parte dos restantes utilizadores da memória, já que em certas ocasiões, o *NIC* tem uma actividade muito intensa sobre a sua memória externa. Optou-se então pelo uso de uma memória *dual-port*. Esta memória permite a utilização em simultâneo por duas entidades, tendo o seu próprio gestor de acesso. O acesso a esta memória só não é permitido se de ambos os lados for acedida no mesmo instante, no mesmo endereço e se pelo menos um dos acessos for de escrita.

CONCLUSÕES

Ainda a meio do projecto nos demos conta do quão volumoso ele era na realidade. Em parte devido à novidade dos assuntos que são focados, ao grande número de circuitos que era necessário desenvolver e ainda à complexidade da topologia que foi por nós seguida, nomeadamente a multiplexagem dos vários blocos. Mesmo assim tentámos realizar o maior número de itens que nos foi possível e, em certas ocasiões, desenvolvemos outros que não eram pedidos mas que nós julgarmos serem indispensáveis tendo em vista a maior funcionalidade da placa. Entre eles encontra-se a possibilidade de comunicação assíncrona entre o *NIC*, *PC* e *transputers* (*interrupts* e *events*).

Uma coisa que notámos foi que os prazos que nós previmos para a conclusão de certas tarefas foram sendo progressivamente aumentados devido a contratempos pontuais e imprevisíveis, o que nos leva a concluir que nestes casos convém dar sempre uma margem de variação para que não surjam surpresas desagradáveis, como seja a falta de tempo para acabar tarefas importantes. Daquilo que nos foi proposto concluímos o que se pode chamar primeira fase do projecto que era a síntese do circuito integrado *VLSI* que pudesse servir de *interface* e árbitro.

O projecto ficou pelo desenvolvimento de um ASIC. Falta começar a segunda e última parte deste projecto que é o desenho da placa de circuito impresso, da qual já fizemos um primeiro desenho completo, mostrando como devem ser interligados todos os elementos.

Em simultâneo pode-se ir pensando num, ou antes vários programas em *OCCAM* de teste do placa quadriprocessador. Esta fase é muito importante e os programas devem testar todo o tipo de situações, por mais insólitas que sejam. Deveria também ser desenvolvido um bom programa de interface com o utilizador para *PC* para tornar a utilidade futura deste projecto uma realidade e não só uma boa ferramenta de ensino para professores e alunos.A totalidade do trabalho que foi desenvolvido foi feito no *Cadence*, com a ajuda dos seus programas satélite e bibliotecas *ECPD10*. Depois de tantos meses, quase um ano, a trabalhar com ele descobrimos algumas particularidades que dependem da maneira particular como cada um prefira trabalhar.

Estamos convictos que o projecto que nos propusemos desenvolver é muito útil e pode vir a ser bastante usado em aplicações que necessitem de grande capacidade de processamento paralelo tais como tratamento e síntese de imagens, de som, controlo de grandes quantidades de informação em tempo real, etc. Se no futuro se juntarem duas ou mais placas quadriprocessador como esta de modo a formarem uma rede seria interessante fazê-las correr um programa do tipo de redes neurais, ou seja tentar desenvolver aplicações tendo em vista a inteligência artificial.

Apesar da pastilha não se encontrar pronta a ser produzida, já que seria necessário proceder a mais algumas simulações, a fim de estudar o seu comportamento em alguns pormenores, foi feito o *place* e *route*. A primeira operação permite posicionar geograficamente na pastilha, as diferentes células que compõem o circuito. Este operação pode ser feita em parte automaticamente. A segunda executa as ligação entre células de modo a implementar o circuito projectado.

Apesar desta fase também poder ser automática, por vezes é necessário efectuar algumas correcções. Na primeira fase é gerado um ficheiro com alguns dados em forma de previsões. Estas permitem ter uma ideia de algumas dimensões da pastilha de silício final. Assim, foi verificado que a pastilha é *pad limited*, ou seja, as suas dimensões são impostas pelo número de pinos e não pela área do circuito. O tamanho deste circuito é cerca de 6.6mm^2 no entanto a área do ASIC incluindo os *pad's* de *I/O* é aproximadamente 34.9mm^2. Um facto importante que se constatou, foi o de a área ocupada pelas macro células utilizadas no esquemático do árbitro, ser bastante grande.

AGRADECIMENTOS

Um agradecimento muito especial para o Engº Velez e o Engº Joaquim Ferreira que se prontificaram a elucidar-nos em todos os problemas referentes ao uso do software e das máquinas disponíveis desbloqueando-nos os meios e conhecimentos necessários de modo a facilitar o nosso trabalho.

REFERÊNCIAS

[1] *The transputer databook* - INMOS

[2] *IBM AT technical reference manual* - IBM

[3] *Cadence VLSI Design Notes* - A.P. Marriott

[4] *ES2 Asic Design Guidelines* - European Standard Structures

[5] *Data sheet MSC2312A -xxYS9/KS9 DRAM* - OKI semiconductor

[6] *Data sheet do DP8390C Network Interface Controller* - National Semiconductor

[7] *Data sheet da IDT7006S/L* - Integrated Device Tecnology, Inc.