INTERFACES PARA TELEVISÃO DIGITAL.

ESTUDO DE ARQUITETURAS E PROJECTO DE HARDWARE.

Trabalho realizado por:

Nuno Fernando Senra Arantes

INESC Instituto de Engenharia
de Sistemas e Computadores

PARECER

Confirmo o empenho revelado no estágio pelo *Nuno Arantes* bem como a qualidade do trabalho realizado e intitulado "***Interfaces para Televisão Digital***".

Considero assim o candidato merecedor da bolsa **PRODEP**.

Porto, 23 de Julho de 1993.

MÁRIO JORGE LEITÃO
Director

Largo Mompilier 22 • Apartado 4433 • 4007 Porto Codex • Portugal • Tel. 351 2 321006 • Fax 351 2 318692 • Tlx. 23023 INESC P
Pessoa Colectiva n.º 501107169 • Associação Privada sem fins lucrativos, declarada de Utilidade Pública • C.R.C. de Lisboa n.º 72

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Departamento de Engenharia Electrotécnica e de Computadores
Rua dos Bragas, 4099 Porto Codex, PORTUGAL
Telef. 351-2-317105/107/412/457 · Telex 27323 FEUP P · Telefax 351-2-319280

PARECER

Estágio PRODEP de
Nuno Fernando Sousa Arantes

O Nuno Arantes desenvolveu um trabalho intitulado "Interfaces para Televisão Digital" com grande dedicação e mérito.

Para além do trabalho de concepção projectou hardware que vai ser implementado. O trabalho é condiderado bom e merecedor da bolsa proposta.

Porto, 23 de Julho de 1993

Artur Pimenta Alves
(Prof. Associado, do DEEC)

1. INTRODUÇÃO

O trabalho realizado no âmbito deste projecto destina-se a ser aplicado na concepção de uma placa para digitalização de imagens de televisão provenientes de uma câmara de vídeo. A referida placa pode ter duas configurações possíveis (figuras 1 e 2).

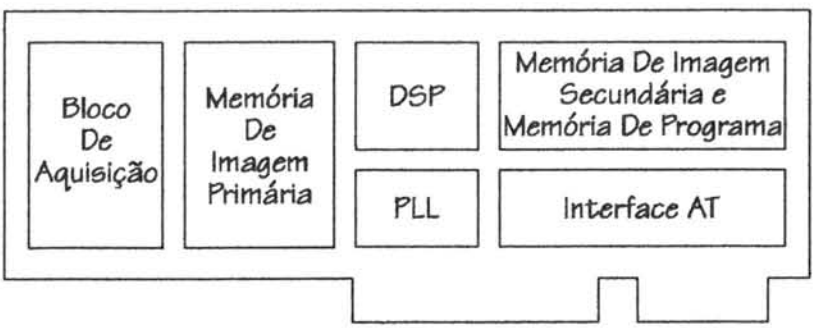


Figura 1

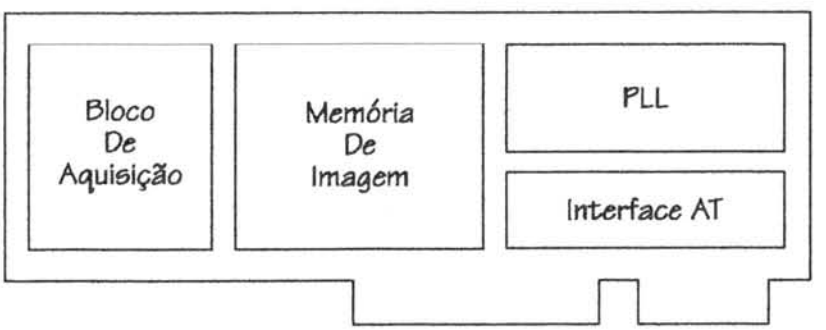


Figura 2

Como podemos ver, em ambos os casos existe um bloco de aquisição que é responsável, entre outras coisas, por fazer a conversão A/D das imagens de vídeo guardando de seguida essa informação na memória de imagem. Foi na concepção desse bloco que consistiu este projecto. Em rigor o bloco de aquisição é constituído pelo bloco desenvolvido no âmbito deste trabalho e por um ADC. No entanto, como o ADC é controlado exclusivamente pelo bloco por nós desenvolvido, referir-nos-emos a este bloco como bloco de aquisição.

No caso da figura 1 essa memória de imagem (memória de imagem primária) é acedida por um processador digital de sinal (DSP). Esse DSP pode tratar então a imagem (aplicando, por exemplo, algoritmos de compressão) e colocar o resultado desse tratamento na memória de imagem secundária. Esta memória contém também o programa usado pelo DSP. A memória de imagem secundária pode também ser acedida pelo PC-AT que aí vai ler a imagem já tratada pelo DSP. O PC-AT pode dar múltiplas aplicações a esta imagem, sendo um exemplo o seu envio pela RDIS para outro local

(isto torna-se particularmente vantajoso, por exemplo, para implementar sistemas de televigilancia em que o centro de controlo não se encontra no mesmo edifício que as câmaras de vídeo).

No caso da figura 2 só existe uma memória de imagem. Essa memória de imagem para além de ser acedida pelo bloco de aquisição também pode ser acedida pelo PC-AT. Este sistema, apesar de mais simples, obriga a que qualquer tratamento levado a cabo na imagem tenha de ser feito pelo PC-AT depois de ter retirado a imagem da memória de imagem existente nesta placa. Nota-se pois uma diferença fundamental de filosofia (ao nível do hardware) entre este sistema e o da figura 1. Assim, o sistema da figura 1 divide a memória de imagem primária apenas entre o bloco de aquisição (que escreve nela) e o DSP (que a lê), e o segundo bloco de memória entre o DSP e o PC-AT(que a lê). A memória de imagem primária é pois invisível para o PC-AT. Por seu lado também o segundo bloco de memória é invisível para o bloco de aquisição. Quanto ao sistema da figura 2, vemos que o único bloco de memória existente é dividido entre o bloco de aquisição e o PC-AT. Constatamos pois que em ambos os sistemas o bloco de aquisição só "vê" um bloco de memória. Este facto faz com que o bloco de aquisição desenvolvido funcione indiferentemente em ambos os sistemas. Isto deve-se à forma de funcionamento deste bloco (que explicaremos mais adiante).

Existem placas em que a aquisição das imagens é controlada pelo processador (no caso do sistema da figura 2 pelo DSP). Se atendermos ao facto de que uma imagem de televisão dura $1/25\ Hz = 0.04\ s$ (o que num processador a trabalhar a 33 MHz corresponde a 1320000 ciclos de clock) vemos é muito penalizante manter um processador parado (controlando apenas a aquisição) durante esse tempo. Claro que numa placa destas por cada imagem adquirida são descartadas várias (tipicamente 5 ou 6) sendo o tempo correspondente a estas imagens descartadas utilizado para tratar a imagem adquirida. Mesmo assim a utilização de um processador para controlar a aquisição implica um aproveitamento pouco eficiente deste podendo levar à impossibilidade de aplicação de algoritmos de compressão de imagem mais complexos. A solução reside no sistema desenvolvido no âmbito deste trabalho. Este sistema realiza a aquisição das imagens de forma totalmente autónoma. Isto é, trata-se de um bloco de hardware que:

- dá ordem ao ADC para digitalizar ou não uma imagem,
- coloca os pixeis válidos (não os correspondentes aos sinais de sincronismo) de imagem na memória de imagem gerando os endereços e o sinal de Write para a referida memória,
- determina em função da resolução pretendida (como veremos à frente) se uma determinada imagem, campo ou linha deve ser adquirida gerando de forma concordante os sinais de actuação da memória e do ADC.

Este bloco permite adquirir imagens de vários formatos. Assim, os formatos possíveis são: 720×576 (norma 601 do CCIR), 360×288 e 180×

144. A escolha do formato pretendido é feita apenas pela actuação de dois sinais de entrada deste bloco. É pois de realçar a simplicidade e a flexibilidade possibilitadas por esta solução.

A primeira hipótese posta para a realização deste bloco foi a utilização de um microcontrolador. Essa hipótese foi no entanto posta de lado devido ao facto de na resolução mais alta (720×576) ser necessário trabalhar com um sinal de clock de $1/72.2222\ ns \approx 13.846\ MHz$ (este valor será explicado mais adiante).

Uma segunda hipótese seria o uso de lógica discreta. No entanto, a grande quantidade de lógica necessária levava a que, mesmo utilizando lógica bastante rápida (TTL da série F), os atrasos se fossem propagando por essa mesma lógica de uma forma incomportável.

Optamos assim pela utilização de uma EPLD (erasable programable logical device). Estes dispositivos trabalham até frequências acima dos 20 MHz não se pondo pois os problemas inerentes às soluções referidas anteriormente. Esta opção revelou-se a mais acertada por vários motivos. Em primeiro lugar a programação destes dispositivos é feita através de um instrumento de CAD bastante poderoso que dá pelo nome de MAXPLUS2. As vantagens na utilização deste instrumento que permite não só a utilização de uma linguagem de alto nível para programar o dispositivo mas também a simulação em computador do seu funcionamento são por demais evidentes. Foi assim possível partir para a implementação física da EPLD já com ela devidamente testada tendo todos os problemas encontrados na fase de simulação sido resolvidos antes da referida implementação. Outra vantagem no uso de uma EPLD consiste na poupança de espaço que este dispositivo permite. De facto, uma implementação baseada em lógica discreta iria ocupar muito espaço numa placa onde o espaço não abunda (lembremos que esta placa se destina a um slot de expansão de um PC-AT). Pelo contrario, a grande compressão obtida com a EPLD permite o uso de DSPs de dimensões razoáveis e até mesmo a colocação de dois DSPs na placa.

2. FORMATOS PERMITIDOS

Tal com já referimos este módulo permite a obtenção de imagens em vários formatos. Para vermos como esses formatos são obtidos torna-se necessário considerar algumas características do sinal de vídeo.

De acordo com a norma do CCIR, um sinal de televisão PAL do tipo B tem as características expostas na Tabela 1.

Analisemos agora a forma de obter os vários formatos. A resolução horizontal é controlada através da frequência do sinal que faz a amostragem

do sinal de vídeo em cada linha. A resolução vertical é obtida controlando

CARACTERÍSTICAS DE UM SINAL PAL DO TIPO B

Número de campos por imagem	2
Número de linhas por imagem	625
Frequência de campo	50 Hz
Frequência de linha	15625 Hz
Duração de uma linha	64 μs
Duração do impulso de sincronismo horizontal	4.7 μs
Duração do "blanking" horizontal	12 μs
Duração do "front porch"	1.5 μs
Duração do "blanking" vertical	25 linhas
Largura de banda	5 MHz
Relação de aspecto da imagem	4/3

Tabela 1

quais os campos e linhas a adquirir.

Comecemos pois pelo formato 720×576. Podemos ver pela Tabela 1 que embora a duração de uma linha seja 64 μs, apenas em 52 μs existe imagem válida. De facto, durante os últimos 12 μs de cada linha são constituídos pelo "blanking" horizontal não se tratando pois de imagem válida. Como pretendemos 720 pixeis por cada linha o período de amostragem será dado por :

$$\frac{52\,\mu s}{720} \approx 72.2222\ ns$$

O que implica uma frequência de amostragem aproximadamente igual a 13.846 MHz. Quanto à resolução vertical podemos ver que das 625 linhas que constituem uma imagem, 50 delas (25 por cada campo) correspondem ao "blanking" vertical. Isto é, por cada imagem temos 575 linhas de imagem válida, ou seja, a resolução vertical pretendida. Devemos pois adquirir todas as linhas dos dois campos para obter a resolução vertical pretendida.

Passemos agora ao formato 360×288. Realizando cálculos similares aos anteriores para a resolução horizontal vemos que o período de amostragem vem dado por:

$$\frac{52\,\mu s}{360} \approx 144.4444\ ns$$

A este período corresponde uma frequência de amostragem aproximadamente igual a 6.923 MHz, isto é, metade da anterior. A resolução vertical é obtida adquirindo todas as linhas de um dos campos da imagem. De facto cada campo contém 312.5 linhas. Se a estas 312.5 linhas retirarmos as 25 linhas desse campo correspondentes ao "blanking" vertical,

obtemos 287.5 linhas de imagem válida (o que é sensivelmente a resolução vertical pretendida).

Finalmente, vejamos o formato 180×144. Cálculos idênticos aos anteriores para a resolução vertical permitem-nos concluir que o período de amostragem deverá ser:

$$\frac{52\,\mu s}{180} \approx 288.8889\,ns$$

Isto implica uma frequência de amostragem sensivelmente igual a 3.4615 MHz (um quarto da obtida para o formato 720×576). Quanto à resolução vertical podemos ver que metade de um dos campos da imagem corresponde a 143.75 linhas de imagem válida (o que é aproximadamente a resolução pretendida). Neste formato devemos pois amostrar linha sim linha não de um dos campos da imagem.

3. BLOCO DE AQUISIÇÃO

O funcionamento em linhas gerais do bloco de aquisição a que este trabalho se refere pode ser descrito com base no diagrama de blocos da Figura 3.

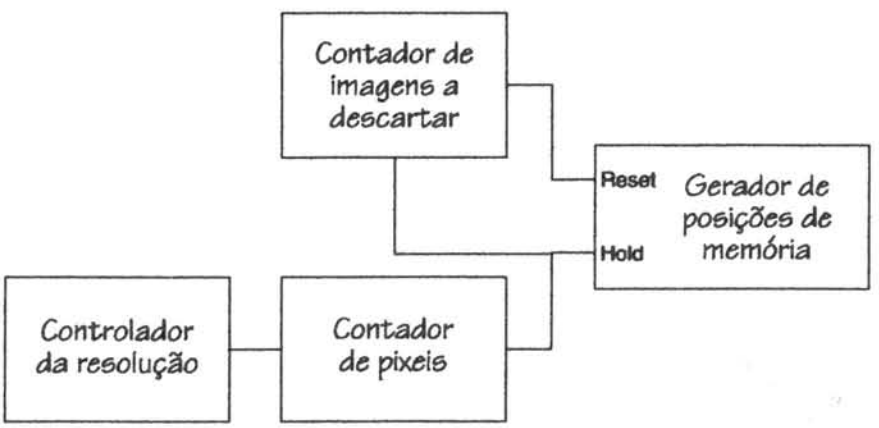


Figura 3

O gerador de posições de memória gera não só as posições de memória como também o sinal de Write para a memória e o sinal de Output Enable para o ADC.

Deve também existir um bloco que conte o número de imagens a descartar e que durante esse período faça o Hold do gerador de posições de memória inibindo assim todas as saídas do sistema. Sempre que este contador concluir que uma dada imagem deve ser adquirida, então, deve ser feito o Reset (e retirado o Hold) do gerador de posições de memória no começo da referida imagem.

Sempre que está a decorrer a aquisição de uma imagem deve ser feito Hold a todo o sistema durante os períodos correspondentes ao "blanking" horizontal. De facto, esses períodos são compostos por um conjunto de sinais que não contêm imagem válida e surgem a separar linhas consecutivas. Existe pois um bloco que a partir do inicio de cada linha conta o número de pixeis que a compõe (que varia consoante o formato pretendido). Quando este bloco conclui que já foram adquiridos todos os pixeis que compõem a linha, faz o Hold o sistema. Quando começar uma nova linha o Hold do sistema é retirado, o contador de pixeis é reinicializado o o processo repete-se.

No entanto, consoante o formato pretendido nem todos os campos e/ou linhas da imagem devem ser adquiridos. Tem pois que existir um bloco (Controlador da resolução) que conforme o formato pretendido dê ordem ao Contador de pixeis para não adquirir uma determinada linha colocando o sistema em Hold. Esta ordem pode-se manter durante um campo inteiro quando o formato pretendido obrigar a que um dos campos não seja adquirido.

Esta descrição do funcionamento deste sistema foi muito simplificada. Assim, comentamos no próximo ponto o diagrama de blocos utilizado no MAXPLUS2 para implementar este sistema.

3.1 Diagrama De Blocos Usado No MAXPLUS2

O diagrama de blocos usado no MAXPLUS2 para implementar este sistema está apresentado do Anexo 1.

Os problemas encontrados na prática para implementar este sistema levaram a um grau de complexidade bastante maior.

Uma das maiores dificuldades encontradas prende-se com o facto de os sinais disponíveis para o controlo do sistema (sincronismo vertical, sincronismo horizontal e campo par/ímpar) serem por natureza assíncronos. Ora, a EPLD é um dispositivo iminentemente síncrono. Como tal muitos dos blocos que a constituem contêm flip-flops. Assim, a natureza assíncrona dos sinais de controlo fazia com que o Setup Time e o Hold Time dos flip-flops não fossem garantidamente respeitados. Isto fazia com que os flip-flops perdessem o seu valor levando a situações de funcionamento anormal. Para as máquinas de estados existentes em alguns blocos era mesmo impossível sair destas situações o que levava ao errado funcionamento de todo o sistema. Todos estes problemas foram observados durante a fase de simulação que, como já referimos, é possível realizar no MAXPLUS2.

A solução encontrada foi tornar os sinais de controlo síncronos com o sinal de clock fazendo-os passar por flip-flops à entrada da EPLD. Com esta solução existe uma incerteza de um ciclo de clock (dependendo do instante em que surgem as transições dos referidos sinais de controlo na entrada da EPLD). No entanto, todo o sistema fica síncrono eliminando os problemas

referidos anteriormente. Esta solução acarreta ainda um outro efeito que tem de ser levado em linha de conta. De facto, mesmo que os sinais de controlo surjam síncronos com o clock na entrada da EPLD (actuando imediatamente os flip-flops colocados na entrada), as saídas dos referidos flip-flops só serão actualizadas no próximo ciclo de clock. Isto é, para além da incerteza de um ciclo de clock referida anteriormente, existe um atraso (também ele de um ciclo de clock) que é garantido. Este atraso fixo pode ser controlado, porém, nos vários contadores existentes nos blocos constituintes do nosso sistema (o que demonstra as vantagens, ao nível da controlabilidade, de termos um sistema inteiramente síncrono).

Comecemos pois por analisar os vários blocos apresentados no Anexo 1.

3.1.1 Bloco "cont_vsy"

Os sinais de sincronismo no inicio de cada campo da imagem são algo complexos. De facto, juntamente com o sinal de sincronismo vertical surgem uma série de outros impulsos denominados Impulsos De Equalização. Estes impulsos servem para melhorar as condições de obtenção do sinal de sincronismo vertical (que é obtido por integração). O conjunto destes sinais está representado na Figura 4.

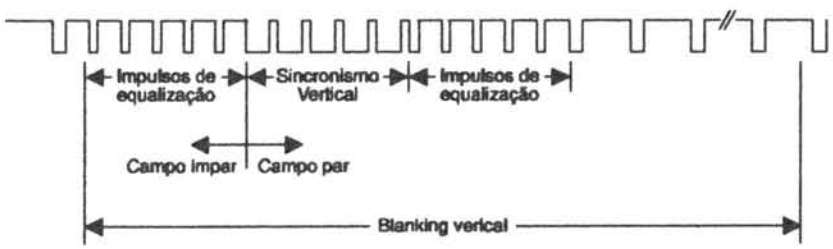


Figura 4

Estes impulsos de equalização poderiam ser tomados por mais sinais de sincronismo vertical pelo nosso sistema. Optamos pois por "fabricar" um sinal de sincronismo vertical com a duração do verdadeiro sinal de sincronismo vertical mais a duração do segundo conjunto de impulsos de equalização. Isto é, um sinal activo baixo que vem abaixo no inicio do sinal de sincronismo vertical e vai acima no final do segundo conjunto de impulsos de equalização. Este sincronismo vertical artificial é usado pelo sistema como sendo o verdadeiro sincronismo vertical. A duração deste sinal é de 320 μs já que tanto o sinal de sincronismo vertical como o segundo conjunto de impulsos de equalização duram 160μs.

A função deste bloco é pois a geração do atrás referido sinal de pseudo-sincronismo vertical.

De notar ainda que os 320 μs de duração do sinal são obtidos à custa da contagem de ciclos de clock (contagem essa que varia conforme a

resolução pretendida). Este processo não produz um sinal com a duração exacta de 320 μs uma vez que este valor não é divisível pelo período do sinal de clock. De facto, para a resolução mais elevada obtém-se um sinal com a duração de 320.04 μs, para a resolução média obtém-se a duração de 320.112 μs e o valor obtido para a resolução mais baixa é 319.392 μs. No entanto, devido à natureza síncrona do nosso sistema estas diferenças são irrelevantes.

A saída deste bloco é colocada à entrada de um NOR juntamente com o sinal indicador do campo da imagem (sinal ***/par_impar*** depois de tornado síncrono pela passagem num flip-flop). À saída deste NOR obtemos pois um sinal activo alto que é similar ao sinal gerado no bloco **"cont_vsy"** mas que é activado apenas em um dos campos da imagem (marca portanto o começo de uma imagem). Este sinal (saída do NOR) é usado para fazer o enable do bloco **"ima_desc"** e o reset do bloco **"ger_end"**.

3.1.2 Bloco "ima_desc"

Este bloco é usado para contar o número de imagens a descartar entre duas imagens a adquirir. Tem pois um sinal de saída (***capt_imag***) activo alto que indica quando é que uma imagem deve ser adquirida. Este sinal de saída está ligado a um pino de saída da EPLD (***adquire***) que serve para fazer o output enable do ADC. Está também ligado a cinco pinos de entrada da EPLD. Quatro deles (***A[3..0]***) servem para especificar quantas imagens devem ser descartadas. O outro pino (***controle***) é usado apenas durante o período de simulação e permite colocar o sinal ***capt_imag*** a 1 independentemente das entradas deste bloco.

Passemos pois a descrever sucintamente o funcionamento deste bloco. Sempre que surge uma nova imagem a entrada ***en*** vem acima (ver sinal de saída do NOR no ponto 3.1.1) e o número de imagens que devem ser descartadas é actualizado. Quando o número de imagens a descartar for 0 é sinal que se deve adquirir essa imagem e o sinal ***capt_imag*** é actuado convenientemente. No começo da imagem seguinte o número de imagens a descartar é inicializado em função do conteúdo de A[3..0] e o processo repete-se.

3.1.3 Bloco "cont_bac"

No fim de cada linha a geração de posições de memória e do sinal de Write para a mesma são inibidos até ao começo da linha seguinte. Isto é levado a cabo porque os sinais que funcionam como separadores entre as linhas (front porch, sincronismo horizontal e back porch) não contêm imagem válida, não sendo portanto lógico nem eficiente para o posterior tratamento da imagem que sejam guardados na memória. Ora o inicio de uma linha é marcado pelo fim do sinal de "back_porch". Este sinal seria pois

de grande utilidade para determinar o inicio de cada linha com a consequente desinibição do processo de aquisição. Infelizmente nem o inicio nem o fim deste sinal são síncronos com o sinal de clock inviabilisando pois o uso deste sinal. No entanto, o inicio (bordo descendente) do sinal de sincronismo horizontal é garantidamente síncrono com o sinal de clock.

Estes factores fizeram com que enveredassemos por uma solução idêntica à adoptada para o sinal de sincronismo vertical (ponto 3.1.1). Isto é, geramos um sinal (***back_***) activo baixo cujo inicio coincide com o inicio do sinal de sinal de sincronismo horizontal e cujo fim coincide com o fim do sinal de "back porch". Este sinal (***back_***) deverá ter uma duração de 10.5 μs. Tal como com o bloco **"cont_vsy"** esta duração é obtida por contagem de ciclos de clock. Essa contagem depende obviamente da resolução pretendida (daí a existência das entradas ***resol[1..0]***) pois a frequência do sinal de clock varia com esta. Também aqui se põe o problema de 10.5 μs não ser divisível pelo período do sinal de clock. Devido a isso, a duração do sinal ***back_*** é 10.512 μs para as duas resoluções mais altas e 10.368 μs para a resolução mais baixa. Todavia, pelos mesmos motivos expostos no ponto 3.1.1 isso não afecta o desempenho do sistema. De notar ainda que na duração deste sinal é feita a compensação dos atrasos (devidos à actuação das saídas dos flip-flops) existentes neste bloco bem como nos blocos **"cont_h"** e **"maq_lin"**. Esta compensação é feita colocando o sinal ***back_*** a 1 três ciclos de clock antes do que seria necessário para se atingir os 10.5 μs de duração daquele sinal. Este bloco tem pois como única função a geração do sinal ***back_*** .

3.1.4 Bloco "cont_h"

Tal como referirmos no ponto anterior deve-se evitar que os sinais de "front porch", sincronismo horizontal e "back porch" (que não contêm imagem válida) sejam adquiridos. Assim, quando se faz a aquisição de uma dada linha contam-se o número da pixeis adquiridos. Quando se constata que já foram adquiridos todos os pixeis dessa linha é porque a informação subsequente diz respeito ao "front porch" e todo o sistema deve ser posto em Hold. Ao iniciar-se uma nova linha é retirado o Hold o sistema sendo o processo reinicializado. Estas operações são realizadas pelo bloco **"maq_lin"** (ponto 3.1.5) e pelo bloco a que se refere este ponto.

A função deste bloco é produzir um sinal (***fim_linha***) activo alto que fique activo desde o momento em que a imagem válida de uma linha termina até que se inicia o sinal de sincronismo horizontal. Isto é, este sinal deve ficar activo sensivelmente durante o "front porch". Em rigor a duração deste sinal não coincide com a do "front porch". Isto deve-se ao facto de na realidade se amostrar a linha a uma frequência levemente superior à requerida para obter os formatos permitidos por este sistema. A título de exemplo podemos dizer que em vez de se usar os 13.8461538 MHz

necessários para a resolução mais alta, é licito usar 13.86 MHz. Isto leva a que haja uma "compressão" da imagem contida numa linha. Consequentemente quando o sinal ***fim_linha*** ficar activo ainda existe uma porção mínima de imagem válida que nunca é adquirida e o "front porch" ainda não se iniciou.

O sinal ***fim_linha*** vem a 0 quando o sinal ***back_*** do bloco "**cont_bac**" (ponto 3.1.3) ficar activo (vier a 0). A contagem do número de pixeis adquiridos recomeça no inicio de cada linha, isto é, quando o sinal ***back_*** ficar inactivo (vier a 1).

3.1.5 Bloco "maq_lin"

A função deste bloco é a partir dos sinais ***back_*** (ponto 3.1.3) e ***fim_linha*** (ponto 3.1.4) produzir um sinal (***para_linha***) activo alto que fica activo desde o começo do sinal ***para_linha*** até ao final do sinal ***back_***. Ou seja, as transições ascendente e descendente de ***para_linha*** são coincidentes com a transição ascendente de ***fim_linha*** e com a transição ascendente de ***back_*** respectivamente. Como vemos este bloco é sensível às transições dos sinais ***back_*** e ***fim_linha*** e não ao seu estado lógico. Por isso, a sua implementação é baseada numa máquina de estados.

Tal como podemos observar, o sinal ***para_linha*** fica activo durante o período de tempo em que o sistema deve permanecer em Hold (e com as suas saídas inibidas) entre duas linhas de imagem consecutivas. Esta é pois a única finalidade deste sinal.

3.1.6 Bloco "res_ver"

Tal como já referimos nem todas as linhas da imagem devem ser adquiridas. Assim, para o formato 720×576 devem ser adquiridas todas as linhas dos dois campos da imagem. Já para o formato 360×288 devem ser adquiridas todas as linhas de apenas um dos campos da imagem. Finalmente, para o formato 180×144 a aquisição deve ser feita com base num esquema "linha sim linha não" de somente um dos campos.

O objectivo deste bloco é pois controlar quais as linhas que devem ser adquiridas. Para tal gera um sinal (***capt_linha***) que é actualizado no inicio de cada linha. Quando este sinal está em VCC significa que a presente linha deve ser adquirida, quando está em GND a linha não deve ser adquirida.

Como vimos o funcionamento deste bloco depende do formato pretendido. Por isso, esse formato é introduzido nas entradas ***resol[1..0]***. Para actualizar a saída no início de cada linha, este bloco usa o sinal ***back_*** (pois este sinal antecede sempre o começo de uma linha). Para controlar os campos da imagem é usado o sinal ***/par_impar*** depois de tornado síncrono pela passagem num flip-flop (ver ponto 3.1.1).

3.1.7 Bloco "ger_end"

Este bloco gera os endereços de memória onde os pixeis que estão a ser digitalizados devem ser armazenados. Esses endereços são obtidos nas saídas ***aq[18..0]***. São usados 19 bits (que permitem endereçar 512K posições de memória) porque na resolução mais alta necessitamos de armazenar 720×576=414720 pixeis (e cada pixel fica numa posição de memória).

Como vimos atrás, há alturas em que a geração de posições de memória deve ser suspensa (posta em Hold) mas não reinicializada. Isto é conseguido pondo um sinal de entrada deste bloco (***hold***) em VCC. Quando ***hold*** está em GND a geração de endereços processa-se normalmente. A este sinal de entrada estão ligados os sinais ***capt_imag***, ***para_linha*** e ***capt_linha*** através de um NAND e de um OR (figura 5).

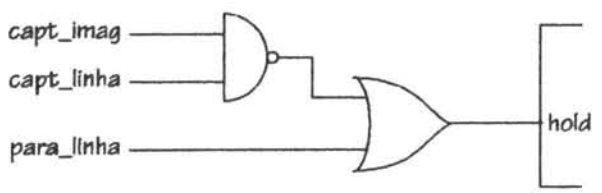


Figura 5

Esta lógica discreta permite que o sinal de entrada ***hold*** seja activado em três situações :

- Quando estamos perante uma imagem que não deve ser adquirida
- Quando estamos perante uma linha que não deve ser adquirida, mas dentro de uma imagem que está a ser digitalizada
- Quando, dentro de uma imagem que está a ser adquirida, nos deparamos com os sinais "front porch", sincronismo horizontal e "back porch" que existem entre duas linhas consecutivas.

É também necessário que cada imagem seja sempre armazenada a partir da posição 00000h. Para tal existe uma entrada (***reset***) activa alta que quando activada provoca o Reset do gerador de endereços. O sinal que está ligado a esta entrada é o que se obtém à saída do NOR discutido no ponto 3.1.1. Este sinal é ideal para este fim uma vez que marca o inicio de cada imagem e termina antes de se iniciar a parte válida da mesma.

Quando não está a ser adquirida nenhuma imagem o DSP ou o PC/AT têm de poder aceder à memória de imagem para lerem a ultima imagem adquirida. É assim necessário que durante esse período de tempo a EPLD liberte o barramento de endereços da memória de imagem colocando as linhas ***aq[18..0]*** no estado de alta impedancia. Este bloco possui uma entrada para esse efeito (***out_en***). Quando o sinal aplicado a esta entrada está a 1, as linhas ***aq[18..0]*** contêm o endereço de memória que está a ser

gerado. Contudo, quando o sinal aplicado a ***out_en*** estiver a 0 as linhas ***aq[18..0]*** são postas no estados de alta impedancia libertando assim o barramento de endereços. O sinal ideal para ser aplicado a ***out_en*** é o sinal ***capt_imag*** (ver ponto 3.1.2) uma vez que está a 1 enquanto se adquire uma imagem e está a 0 quando não está a ser feita qualquer aquisição.

3.1.8 O sinal de Write para a memória

Este sistema foi pensado para trabalhar com memórias estáticas (devido às elevadas frequências a que funciona). Especificamente, as memórias a que se destina são as MTC1008 (com tempo de acesso de 20 ns) da Micron. Estas memórias necessitam de um sinal de Write activo baixo com 11 ns (no minimo) de duração. Os dados são guardados nas memórias no final deste sinal (bordo ascendente). Por outro lado não convém gerar este sinal antes de os endereços estarem estaveis na linhas ***aq[18..0]***. Durante a fase de simulação constatamos que o atraso na geração dos endereços em relação ao sinal de clock é de 13 ns. Quanto ao ADC que se destina a ser usado com este sistema (BT253 da Brooktree), demora 40 ns a digitalizar um pixel.

Convém-nos pois um sinal de Write que:

- fique activo (GND) não antes do que 13 ns depois do inicio do ciclo de clock,
- dure pelo menos 11ns,
- permaneça activo pelo menos até 40 ns depois do inicio do ciclo de clock
- fique inactivo antes se passarem 13 ns a partir do inicio do proximo ciclo de clock.

O próprio sinal de clock é pois um sério candidato ao sinal de Write. Seria no entanto aconcellhavel que este sinal terminasse antes do inicio do ciclo de clock seguinte. Isto é, convinha-nos um sinal igual ao sinal de clock mas levemente adiantado em relação a este. Dada a periodicidade do sinal de clock, adiantar este sinal é equivalente a atrasa-lo mais do que meio ciclo.

Existem dois elementos que provocam o atraso atrás referido. Um desses elementos é um NOR pelo qual o sinal de clock atrasado tem de passar (as razões que levam à existencia deste NOR são explicadas mais abaixo). Durante o periodo de simulação verificamos que o atraso provocado por este elemento era sensivelmente igual a meio ciclo. O sinal obtido à saida deste NOR era pois aproximadamente igual ao clock negado. Esta solução não era viavel pois para a resolução mais alta (periodo $\approx$ 72 ns) o sinal de Write viria acima antes de passados 40 ns desde o inicio do ciclo de clock (vimos acima que é obrigatório que assim não seja). Era pois necessário obter um atrazo adicional (mas que fosse inferior a metade de um ciclo de clock (pois de contrário haveria o perigo de o sinal de Write ainda se

encontrar activo quando fosse gerado um novo endereço). O MAXPLUS2 permite a introdução de um elemento que dá pelo nome de expansor ("exp" no diagrama de blocos do Anexo 1) e cuja única função é atrasar um sinal cerca de 11 ns. Optamos pois por fazer passar o sinal de clock por um destes elementos ficando a saída do expansor ligada ao atrás referido NOR. O atraso obtido pela conjugação destes dois elementos foi tal que obtivemos o sinal de Write com as caracteristicas pretendidas.

O sinal de Write só pode ficar activo quando estiverem a ser digitalizados pixeis. Ou seja, tem que haver um processo de colocar este sinal permanentemente em VCC quando não se estiver a adquirir imagem. Um sinal ideal para esse fim é o existente na entrada ***hold*** do bloco **"ger_end"**. Este sinal vem a 1 sempre que não se está a efectuar aquisição de imagem. Se à entrada de um NOR colocarmos este sinal juntamente com o sinal obtido à saída do expansor, obtemos um sinal de Write com as caracteristicas desejadas. Fica assim explicada a função do NOR acima referido. A saída deste NOR está ligada a um dos pinos de saída da EPLD (***/wr***), constituindo o sinal obtido neste pino o sinal de Write para a memória. Este sinal pode ser observado no Anexo 2 onde é apresentado o resultado de uma simulação realizada pelo MAXPLUS2.

3.1.9 Bloco "conf_adc"

Já dissemos acima que este sistema foi idealizado para trabalhar com o ADC BT253 da Brootree. Este ADC permite várias configurações possiveis. De facto, pode-se definir qual a entrada (possui duas) a digtalizar, qual o sinal que deve ser usado como fonte de sincronismo, qual o formato (em número de bits por cor) que deve ser usado, etc. Esta escolha de configuração é feita escrevendo uma palavra de 8 bits num registo (registo de comando) que o ADC tem para este efeito.

Partindo do principio que o ADC é usado sempre com a mesma configuração, a palavra de 8 bits atrás referida nunca varia e portanto pode ser implementada ligando os pinos correspondentes ao registo de comando do ADC aos niveis de tensão adequados.

Assim, é apenas necessário gerar um sinal de Write (com a duração de pelo menos 50 ns) para o ADC. Este sinal de Write faz com que o ADC leia o conteudo do registo de comando e se configure em função dele.

Esta inicialização do ADC deve ser feita aquando do arranque do sistema.

Também no arranque do sistema é necessário fazer o Reset de vários dos blocos que o compõem. Este facto pode ser aproveitado para fazer a inicialização do ADC ao mesmo tempo que se faz o Reset dos referidos blocos. Estas são pois as funções realizadas pelo bloco **"conf_adc"**.

Este bloco (**"conf_adc"**) tem um único sinal de entrada (***reset***) que está ligado um dos pinos de entrada da EPLD. O sinal de Reset presente neste

pino deve ser activo alto. As saídas deste bloco são o sinal de Write para o ADC (***/wr_adc***) e o sinal de Reset (***/reset***) para os blocos **"cont_bac"**, **"cont_vsy"**, **"cont_h"** e **"maq_lin"**.

O sinal ***/wr_adc*** (que está ligado a um pino de saída da EPLD com o mesmo nome) é activo baixo e tem a duração de um ciclo de clock. Este sinal fica activo no ciclo subsequente àquele em que surgiu o sinal ***reset***. Passado um ciclo de clock desde o instante em que ficou activo, o sinal ***/wr_adc*** é novamente desactivado (posto a VCC). Um ciclo de clock é mais do que suficiente uma vez que a sua duração é de pelo menos 72 ns (no caso da resolução mais alta). A duração é pois garantidamente superior aos 50 ns requeridos (no minimo).

O sinal ***/reset*** é activo baixo ficando activo (a GND) no ciclo seguinte àquele em que surgiu o sinal ***reset***. Quando o ***reset*** fica inactivo (GND), o sinal ***/reset*** fica também inactivo (a VCC) no ciclo seguinte. Podemos pois ver que este sinal (***/reset***) tem sensivelmente a mesma duração de ***reset*** mas está levemente atrasado em relação a este e é activo baixo.

Os sinais ***reset***, ***/reset*** e ***/wr_adc*** podem ser observados no Anexo 3 onde figura o resultado de uma das simulações realizadas pelo MAXPLUS2.

3.2 Estrutura Interna Da EPLD

A estrutura interna da EPLD utilizada (EPM5128 da Altera) está representada na figura 6.

O MAXPLUS2 permite-nos definir quais são macrocélulas (MCELL) usadas pelos elementos constituintes dos blocos do Anexo 1. Esta definição da estrutura interna da EPLD é determinante para os atrasos existentes nas saídas do integrado. De facto, se um bloco estiver distribuído por vários "LABs" (tendo os sinais de percorrer a PIA para passarem de um bloco a outro) é natural que o atraso nas suas saídas seja superior ao que existiria no caso de o referido bloco ter sido implementado em um único "LAB".

É pois necessário tentar várias implementações internas possíveis (sendo este um processo iminentemente iterativo) para se encontrar a solução óptima (a que garante os menores atrasos possíveis).

Esta definição da estrutura interna da EPLD é feita num ficheiro de texto que o MAXPLUS2 procura sempre que compila um projecto. A versão final do referido ficheiro pode ser consultada no Anexo 4.

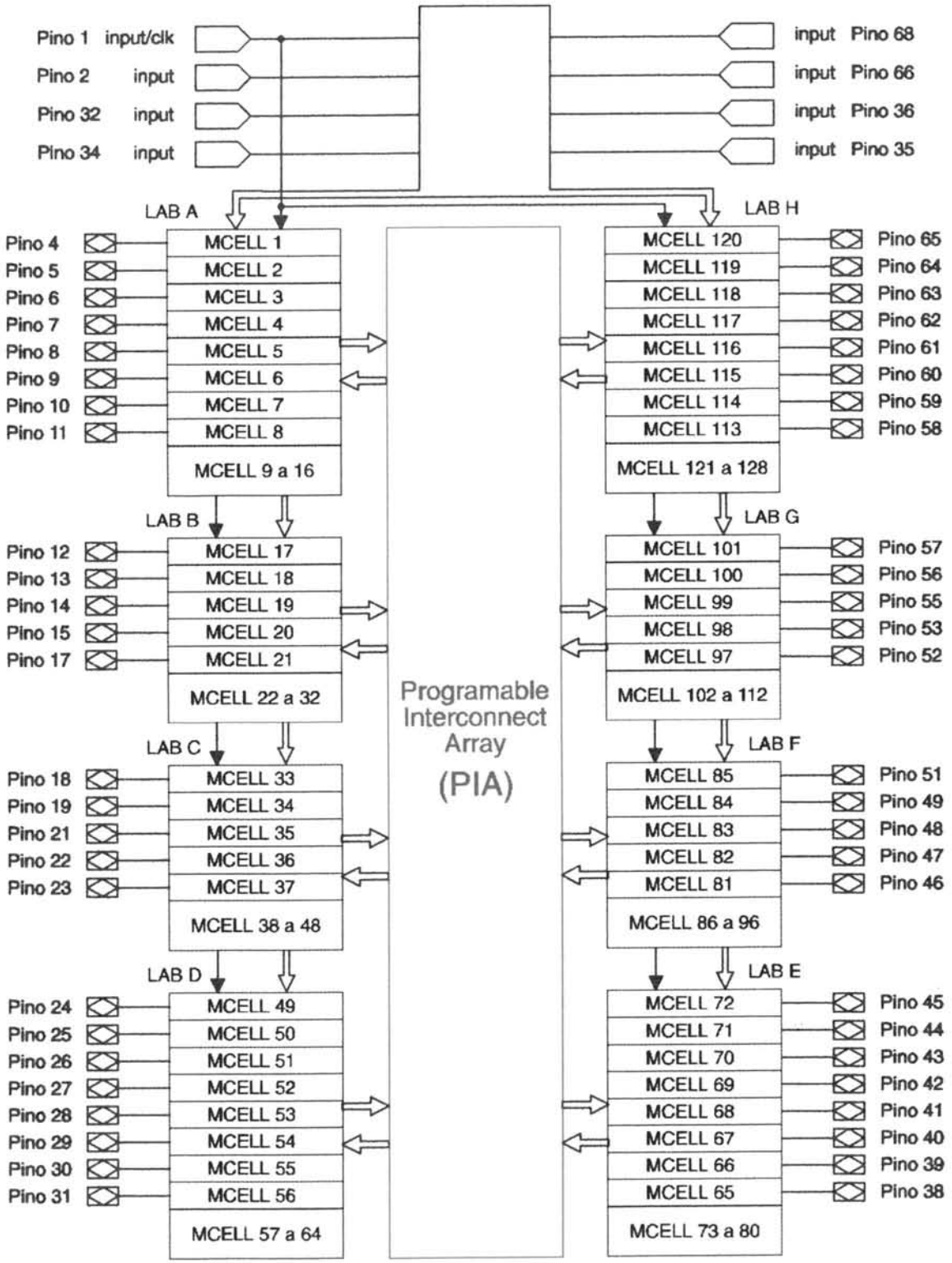
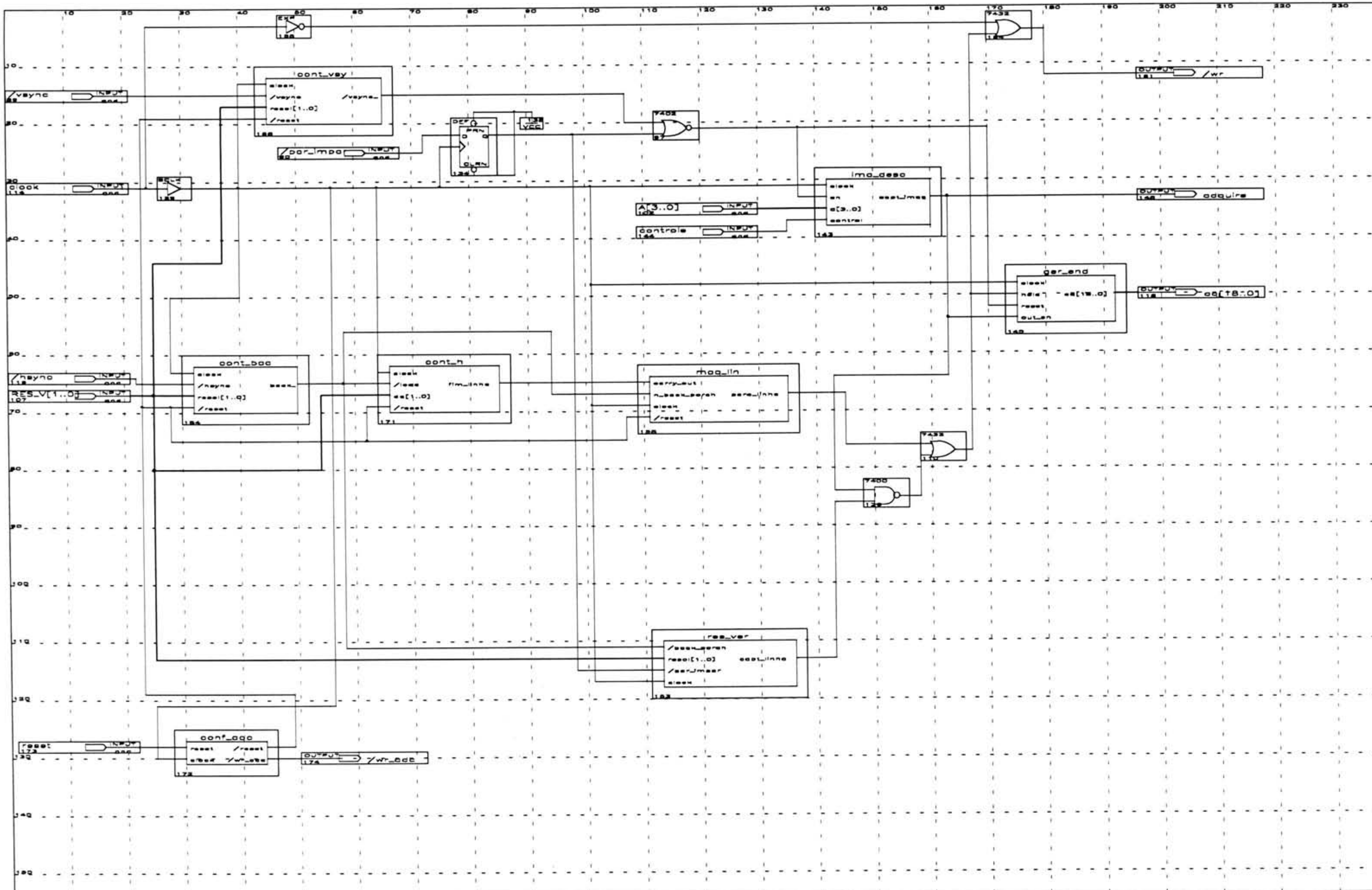

Pino 1 input/clk
Pino 2 input
Pino 32 input
Pino 34 input
input Pino 68
input Pino 66
input Pino 36
input Pino 35
LAB A
LAB H
Pino 4 MCELL 1
Pino 5 MCELL 2
Pino 6 MCELL 3
Pino 7 MCELL 4
Pino 8 MCELL 5
Pino 9 MCELL 6
Pino 10 MCELL 7
Pino 11 MCELL 8
MCELL 9 a 16
MCELL 120 Pino 65
MCELL 119 Pino 64
MCELL 118 Pino 63
MCELL 117 Pino 62
MCELL 116 Pino 61
MCELL 115 Pino 60
MCELL 114 Pino 59
MCELL 113 Pino 58
MCELL 121 a 128
LAB B
LAB G
Pino 12 MCELL 17
Pino 13 MCELL 18
Pino 14 MCELL 19
Pino 15 MCELL 20
Pino 17 MCELL 21
MCELL 22 a 32
MCELL 101 Pino 57
MCELL 100 Pino 56
MCELL 99 Pino 55
MCELL 98 Pino 53
MCELL 97 Pino 52
MCELL 102 a 112
Programable Interconnect Array (PIA)
LAB C
LAB F
Pino 18 MCELL 33
Pino 19 MCELL 34
Pino 21 MCELL 35
Pino 22 MCELL 36
Pino 23 MCELL 37
MCELL 38 a 48
MCELL 85 Pino 51
MCELL 84 Pino 49
MCELL 83 Pino 48
MCELL 82 Pino 47
MCELL 81 Pino 46
MCELL 86 a 96
LAB D
LAB E
Pino 24 MCELL 49
Pino 25 MCELL 50
Pino 26 MCELL 51
Pino 27 MCELL 52
Pino 28 MCELL 53
Pino 29 MCELL 54
Pino 30 MCELL 55
Pino 31 MCELL 56
MCELL 57 a 64
MCELL 72 Pino 45
MCELL 71 Pino 44
MCELL 70 Pino 43
MCELL 69 Pino 42
MCELL 68 Pino 41
MCELL 67 Pino 40
MCELL 66 Pino 39
MCELL 65 Pino 38
MCELL 73 a 80

Figura 6

ANEXO 1

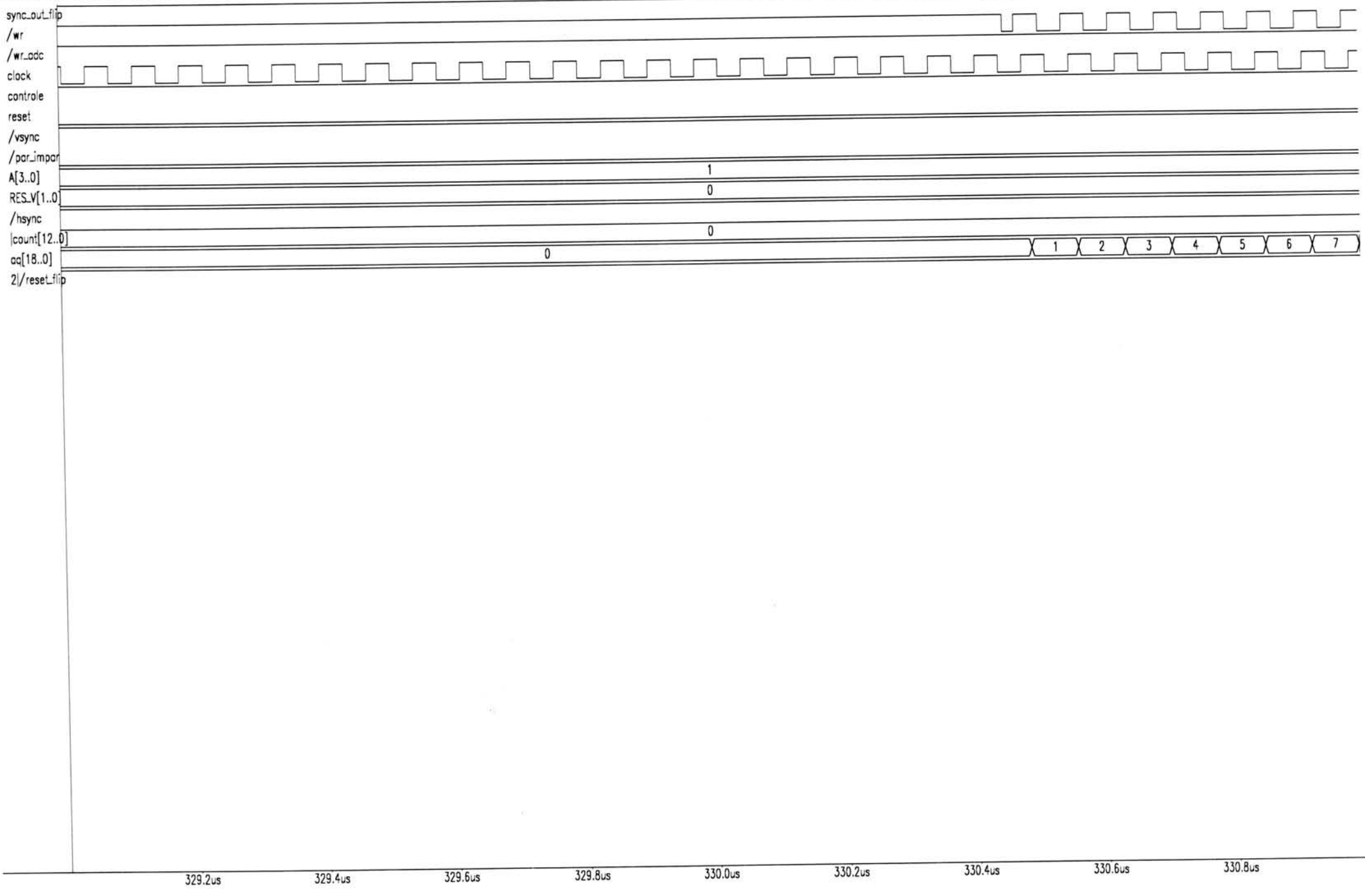
Diagrama de blocos do sistema



cont_vsy
/vsync
clock
A[3..0]
controle
ger_end
cont_bco
/hsync
RES_V[1..0]
cont_h
res_ver
conf_ado
reset
/wr
/wr_adc
adquire
7402
7432
7400
VCC
INPUT
OUTPUT

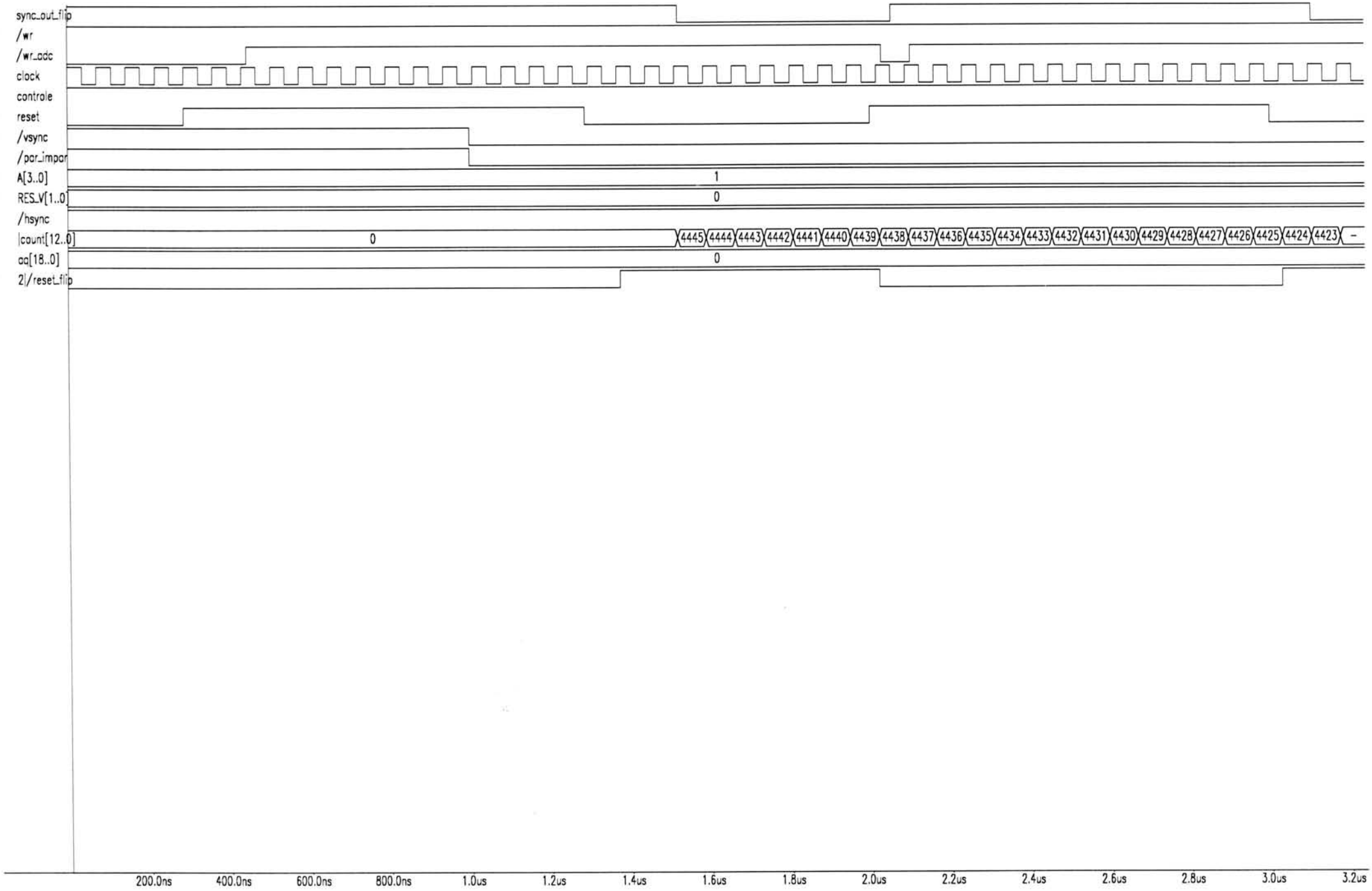
ANEXO 2

Sinal de Write para a memória


sync_out_flip
/wr
/wr_adc
clock
controle
reset
/vsync
/par_impar
A[3..0]
RES_V[1..0]
/hsync
|count[12..0]
aq[18..0]
2|/reset_flip
1
0
0
0
1
2
3
4
5
6
7
329.2us
329.4us
329.6us
329.8us
330.0us
330.2us
330.4us
330.6us
330.8us

ANEXO 3

Sinal de Reset para os vários blocos do sistema e sinal de Write para o ADC


sync_out_flip
/wr
/wr_adc
clock
controle
reset
/vsync
/par_impar
A[3..0]
1
RES_V[1..0]
0
/hsync
|count[12..0]
0
4445
4444
4443
4442
4441
4440
4439
4438
4437
4436
4435
4434
4433
4432
4431
4430
4429
4428
4427
4426
4425
4424
4423
aa[18..0]
0
2|/reset_flip
200.0ns
400.0ns
600.0ns
800.0ns
1.0us
1.2us
1.4us
1.6us
1.8us
2.0us
2.2us
2.4us
2.6us
2.8us
3.0us
3.2us

ANEXO 4

Ficheiro de definição da estrutura interna da EPLD

```
DESIGN IS "aquis"
BEGIN
        DEVICE "aquis" IS "EPM5128-1"
        BEGIN
                A0                              @ 46        : INPUT ; %MC81%
                A1                              @ 47        : INPUT ; %MC82%
                A2                              @ 48        : INPUT ; %MC83%
                A3                              @ 64        : INPUT ; %MC119%
                clock                           @  1        : INPUT ;
                controle                        @ 17        : INPUT ; %MC21%
                /hsync                          @  2        : INPUT ;
                /par_impar                      @ 34        : INPUT ;
                reset                           @ 68        : INPUT ;
                RES_V0                          @ 18        : INPUT ; %MC33%
                RES_V1                          @ 19        : INPUT ; %MC34%
                /vsync                          @ 32        : INPUT ;
                adquire                         @ 49        : OUTPUT ; %MC84%
                aq0                             @ 38        : OUTPUT ; %MC65%
                aq1                             @ 39        : OUTPUT ; %MC66%
                aq2                             @ 40        : OUTPUT ; %MC67%
                aq3                             @ 41        : OUTPUT ; %MC68%
                aq4                             @ 42        : OUTPUT ; %MC69%
                aq5                             @ 43        : OUTPUT ; %MC70%
                aq6                             @ 44        : OUTPUT ; %MC71%
                aq7                             @ 45        : OUTPUT ; %MC72%
                aq8                             @ 52        : OUTPUT ; %MC97%
                aq9                             @ 53        : OUTPUT ; %MC98%
                aq10                            @ 55        : OUTPUT ; %MC99%
                aq11                            @ 56        : OUTPUT ; %MC100%
                aq12                            @ 57        : OUTPUT ; %MC101%
                aq13                            @ 58        : OUTPUT ; %MC113%
                aq14                            @ 59        : OUTPUT ; %MC114%
                aq15                            @ 60        : OUTPUT ; %MC115%
                aq16                            @ 61        : OUTPUT ; %MC116%
                aq17                            @ 62        : OUTPUT ; %MC117%
                aq18                            @ 63        : OUTPUT ; %MC118%
                /wr_adc                         @ 22        : OUTPUT ; %MC36%
                /wr                             @ 23        : OUTPUT ; %MC37%
                |conf_adc:172|q1                @ MC48 : BURIED ;
                |conf_adc:172|/reset_flip        @ MC34 : BURIED ; %PIN 19%
                |cont_bac:154|back_flip          @ MC38 : BURIED ;
                |cont_bac:154|count0             @ MC39 : BURIED ;
                |cont_bac:154|count1             @ MC40 : BURIED ;
                |cont_bac:154|count2             @ MC41 : BURIED ;
```

```
|cont_bac:154|count3                @ MC42 : BURIED;
|cont_bac:154|count4                @ MC43 : BURIED;
|cont_bac:154|count5                @ MC44 : BURIED;
|cont_bac:154|count6                @ MC45 : BURIED;
|cont_bac:154|count7                @ MC46 : BURIED;
|cont_bac:154|q1                    @ MC47 : BURIED;
|cont_h:171|count0                  @ MC9  : BURIED;
|cont_h:171|count1                  @ MC10 : BURIED;
|cont_h:171|count2                  @ MC11 : BURIED;
|cont_h:171|count3                  @ MC12 : BURIED;
|cont_h:171|count4                  @ MC13 : BURIED;
|cont_h:171|count5                  @ MC14 : BURIED;
|cont_h:171|count6                  @ MC15 : BURIED;
|cont_h:171|count7                  @ MC16 : BURIED;
|cont_h:171|count8                  @ MC22 : BURIED;
|cont_h:171|count9                  @ MC23 : BURIED;
|cont_h:171|count10                 @ MC24 : BURIED;
|cont_h:171|count11                 @ MC25 : BURIED;
|cont_h:171|fim_flip                @ MC26 : BURIED;
|cont_vsy:166|count0                @ MC49 : BURIED; %PIN 24%
|cont_vsy:166|count1                @ MC50 : BURIED; %PIN 25%
|cont_vsy:166|count2                @ MC51 : BURIED; %PIN 26%
|cont_vsy:166|count3                @ MC52 : BURIED; %PIN 27%
|cont_vsy:166|count4                @ MC53 : BURIED; %PIN 28%
|cont_vsy:166|count5                @ MC54 : BURIED; %PIN 29%
|cont_vsy:166|count6                @ MC55 : BURIED; %PIN 30%
|cont_vsy:166|count7                @ MC56 : BURIED; %PIN 31%
|cont_vsy:166|count8                @ MC57 : BURIED;
|cont_vsy:166|count9                @ MC58 : BURIED;
|cont_vsy:166|count10               @ MC59 : BURIED;
|cont_vsy:166|count11               @ MC60 : BURIED;
|cont_vsy:166|count12               @ MC61 : BURIED;
|cont_vsy:166|q1                    @ MC62 : BURIED;
|cont_vsy:166|vsync_in_flip         @ MC63 : BURIED;
|cont_vsy:166|vsync_out_flip        @ MC64 : BURIED;
|ima_desc:143|capt_flip             @ MC86 : BURIED;
|ima_desc:143|count0                @ MC87 : BURIED;
|ima_desc:143|count1                @ MC88 : BURIED;
|ima_desc:143|count2                @ MC89 : BURIED;
|ima_desc:143|count3                @ MC90 : BURIED;
|ima_desc:143|q0                    @ MC91 : BURIED;
|maq_lin:158|linha_flip             @ MC92 : BURIED;
|maq_lin:158|q1                     @ MC93 : BURIED;
|maq_lin:158|q2                     @ MC94 : BURIED;
```

```
            |res_ver:153|q1                                   @MC95 : BURIED;
            |res_ver:153|saida_flip                           @MC96 : BURIED;
            :134                                              @MC73 : BURIED;
    END;
END;
```