13. SINCRONIZAÇÃO E INTERACÇÃO COM DISPOSITIVOS EXTERNOS

Dada a estrutura do PDS8_V2, com um único espaço de endereçamento, subentende-se que este espaço irá ser povoado por diferentes tipos de dispositivos. Teremos dispositivos de memória não volátil (ROM, flash, etc.) para conter código, dispositivos de memórias suportando leitura e escrita (SRAM, DRAM, etc.) para conter dados e ainda a existência de dispositivos que permitam a entrada e saída de dados para o exterior do sistema. Esta multiplicidade de dispositivos e interacções implica que o processador tenha mecanismos de sincronização, que lhe permita adaptar-se as especificidades temporais de cada um dos dispositivos sem comprometer o desempenho global do sistema.

13.1 Estados de espera

A inserção de um dispositivo lento (TOE e TWP muito grande) no sistema não deve implicar uma velocidade uniforme do CPU tal que a temporização desse dispositivo seja satisfeita. Com este objectivo iremos adicionar ao PDS8_V2 uma entrada denominada RDY (*Ready*) que, quando activa, significa que o CPU poderá finalizar a acção de leitura ou escrita em curso. Este sinal será testado pelo CPU antes de finalizar o ciclo de leitura ou escrita e, caso este se encontre desactivo, o CPU entra em estado de espera (***Wait state***), prolongando indefinidamente o ciclo em curso até que o sinal RDY se torne activo. No diagrama temporal da Figura 13-1 é mostrado o instante de observação do sinal RDY e a respectiva consequência no comportamento do bus do CPU.

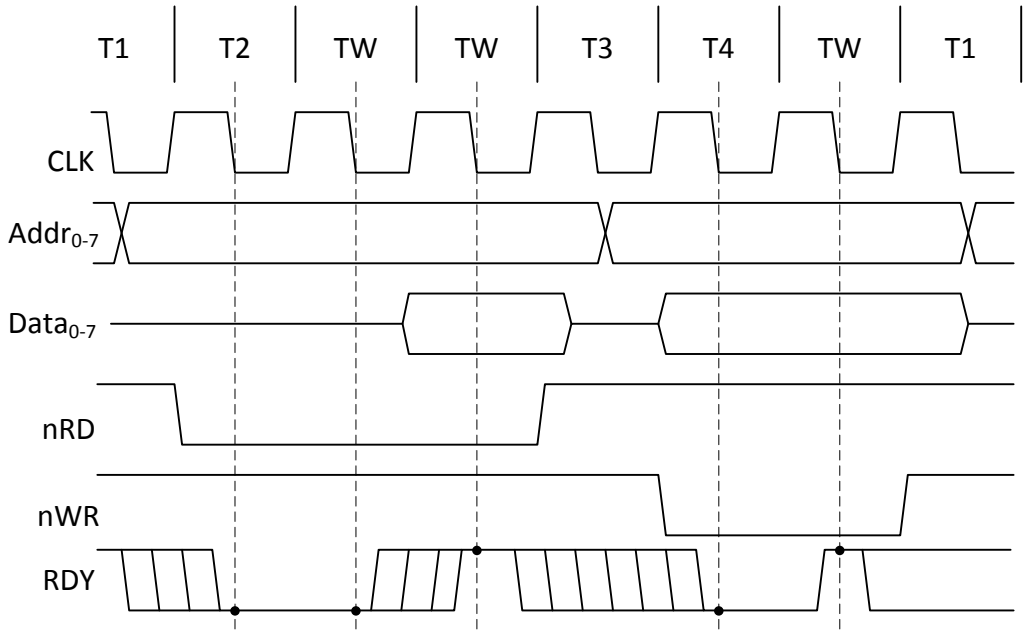

Figura 13-1 Diagrama temporal do sinal de entrada RDY

Para além da situação do sistema poder ser constituído por dispositivos com tempos de acesso muito díspares, este sinal também se torna indispensável num cenário onde o sistema de computação é constituído por vários CPUs com recursos próprios mas partilhando um bus comum para acesso a dispositivos do sistema. Neste cenário pode acontecer que um qualquer CPU ao tentar

aceder ao bus comum, esteja em curso um acesso por parte de outro CPU. Esta situação leva a que o elemento que realiza a arbitragem dos acessos ao bus comum, desactive o sinal de RDY do CPU que pretende aceder, até à conclusão da transferência em curso.

O sinal RDY também é comummente utilizado pelo projectista de sistemas para detectar e corrigir erros de concepção ou de implementação tanto de hardware como de software. Como se pode observar na figura acima, ao desactivar a entrada RDY, o CPU como que cristaliza os buses de endereço, dados e controlo, permitindo ao projectista verificar de forma estática qual o comportamento da lógica associada ao dispositivo que está a ser endereçado e qual o valor do dado que está a ser transferido. Se for construído um circuito que ciclo a ciclo máquina desactive o sinal de RDY, podemos seguir a execução do CPU passo a passo.

13.2 Partilha de BUS

Um outro cenário comum nos sistemas reais, que tem como objectivo diminuir o tempo de transferência de dados entre um dispositivo de I/O e a memória, é a transferência por DMA (*Direct Memory Acess*), ou seja, o sistema dispõe de um dispositivo de DMA que utilizando o bus do CPU realiza a transferência de informação entre o dispositivo de I/O e a memória, sem intervenção do CPU. Nesta situação, o CPU ao receber um pedido do dispositivo de DMA, liberta os *buses* de endereço, dados e controlo (nRD e nWR), colocando-os em alta impedância e entra em estado de espera passiva (***Hold*),** informando simultaneamente o dispositivo DMA que a transferência já se pode realizar. A libertação do BUS, ou seja, a entrada do CPU em estado de *hold*, só pode acontecer quando o ciclo de acesso à memória que por ventura esteja a decorrer tenha sido concluído, o que implica, que o dispositivo requerente do bus, aguarde por sinalização do CPU para iniciar a transferência. Para implementar esta funcionalidade é necessário adicionar ao CPU dois sinais, um de entrada, denominado BRQ (***B**us **R**e**Q**uest*) que o dispositivo requerente activa quando pretende realizar uma transferência e, outro de saída, denominado BGT (***B**us **G**ran**T***), que o CPU utiliza para informar o sistema requerente que o bus já foi libertado. O CPU permanecerá em estado de *hold* enquanto o sinal BRQ estiver activo e durante este tempo o CPU manterá activo o sinal BGT.

Uma vez que um pedido de BRQ só é satisfeito quando o ciclo em curso for concluído, se o CPU estiver em estado de *Wait* (RDY desactivo), só indicará BGT quando for libertado pelo sinal RDY e concluir o ciclo que estava em curso. Em caso de pedido *wait* e *hold* simultâneo, dado que o teste a estes sinais é feito após se ter iniciado o ciclo de leitura ou escrita, o sinal RDY tem prioridade sobre o sinal BRQ, caso contrário implicava interromper um dos ciclos.

No diagrama temporal da **Error! Reference source not found.**Figura 13-2 é mostrado o instante de observação do sinal BRQ e a respectiva consequência no comportamento do CPU.

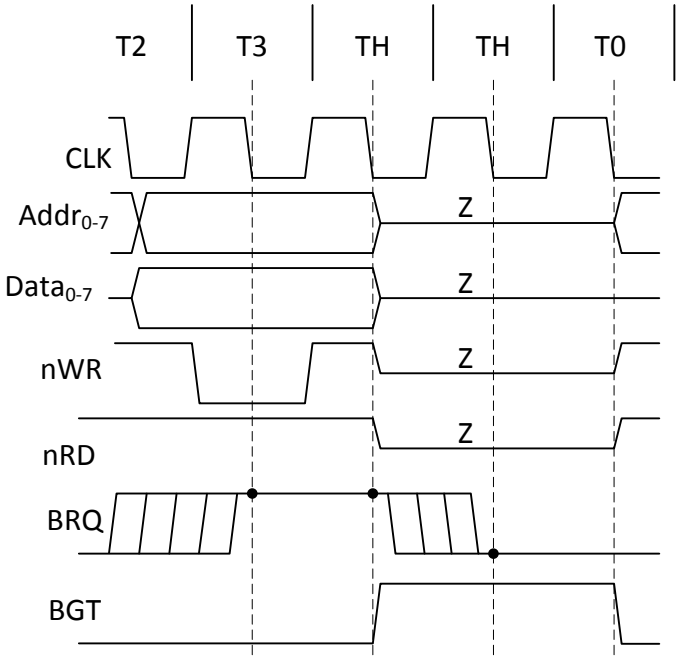

Figura 13-2 Diagrama temporal dos sinais BRQ e BGT

A adição destes três sinais RDY, BRQ e BGT no PDS8_V2 tem as seguintes implicações na estrutura:

- adição de um registo para sincronização dos sinais RDY e BRQ com o flanco descendente de *clock*, afim de garantir a estabilidade dos sinais aquando do teste por parte do módulo de controlo do CPU.

- Inserção de um *flip-flop* para sincronizar o sinal BGT com o flanco descendente do sinal de *clock*, para que a transferência só se inicie a meio de TH, ou seja, com o ciclo de acesso completamente concluído, como é mostrado na figura acima.

Com a introdução destes novos sinais, obtém-se para o PDS8_V2 o esquema bloco mostrado na Figura 13-3.

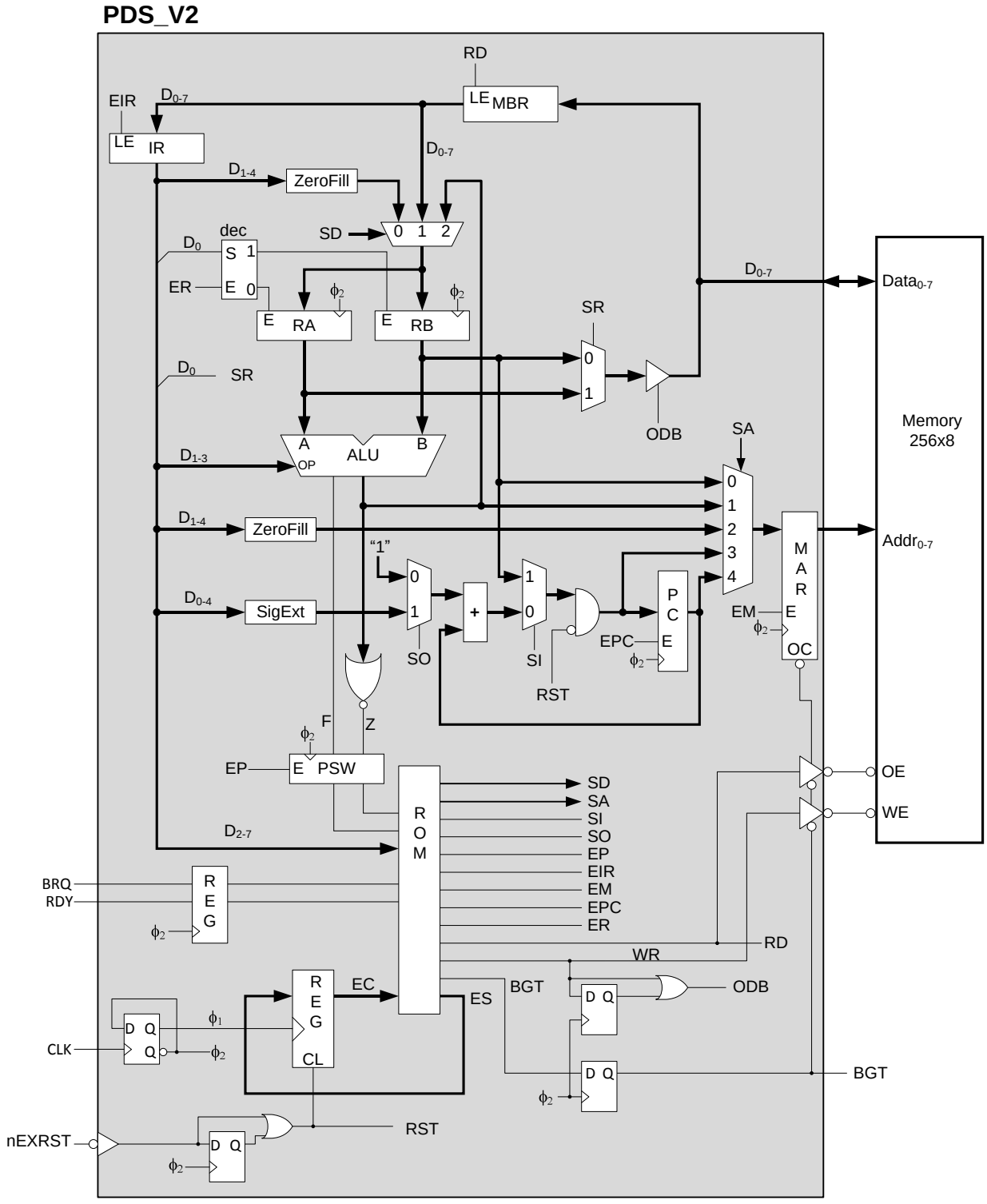

Figura 13-3

Ao nível do módulo de controlo, implica a adição de mais 3 estados, passando o módulo de controlo a ter o comportamento descrito pelo em WinCUPL que a seguir se decreve.

```
sequence[Q0..Q2] {
present Reset
      out EM, EPC, MarMux4;
      next InstFetch;
present AddrFetch_ExecFini
      out EM;
      if (LD # AL) out ER, MarMux4;
      if (JP) out EPC, MarMux3;
      next InstFetch;
present InstFetch
      out RD, EPC;       /* fetch opcode e PC */
      if (!RDY) next WaitFetch;
      if (RDY & BRQ) next HoldFetch;
      if (RDY & !BRQ & (LM # ST)) next MemAccess;
      default next AddrFetch_ExecFini;
present MemAccess
      out EM;
      if (RA) MarMux0;         /* load/Store R,[RB] */
      if (RARB) MarMux1;       /* load R,[RA+RB] */
      if (direct) MarMux2;     /* load/Store R,addr */
      next DataTransf;
present DataTransf
      if (LM) out RD;
      if (ST) out WR;
      if (!RDY) next DataTransf;     /* wait exec */
      if (RDY & BRQ) next HoldExec;
      if (RDY & !BRQ) next AddrFetch_ExecFini;
present WaitFetch
      out RD;
      if (!RDY) next WaitFetch;
      if (RDY & BRQ) next HoldFetch;
      if (RDY & !BRQ & (LM # ST)) next MemAccess;
      default next AddrFetch_ExecFini;
present HoldFetch
      out BGT;
      if (BRQ) next HoldFetch;
      if (!BRQ & (LM # ST)) next MemAccess;
      default next AddrFetch_ExecFini;
present HoldExec
      out BGT;
      if (BRQ) next HoldExec;
      if (!BRQ) next AddrFetch_ExecFini;
}
```