FEUP
Universidade do Porto
Faculdade de Engenharia

Departamento de Engenharia Electrotécnica e de Computadores

Projeto de Sistemas Digitais 2016/2017

4º ano, 1º semestre
Exame 17 de janeiro de 2017

Duração máxima: 3h00m, sem consulta.

[1.5 valores]

1 - Explique como se pode (e deve!) quantificar a qualidade de um processo de simulação funcional do modelo RTL de um sistema digital e explique como tirou partido disso (ou, se não o fez, como deveria ter feito) durante a realização do último projeto laboratorial realizado em PSDi.

[2.5 valores]

2 - Durante o desenvolvimento de um sistema digital integrado seguindo a metodologia de projeto praticada em PSDi, devem ser realizadas vários processos de verificação em diferentes etapas do projeto. Dois desses passos consistem na verificação, por simulação lógica, dos modelos do circuito em projeto antes (simulação funcional) e depois (simulação pós-síntese ou *post-translate*) de realizar o processo de síntese RTL.

[1 valor]

a) Explique em que diferem os modelos que são simulados nessas duas etapas de verificação.

[1.5 valores]

b) Se a verificação pós-síntese já permite verificar a correção funcional do circuito que está a ser projetado, explique por que razão é fundamental realizar o passo de verificação anterior (verificação funcional), apenas na perspetiva do processo de verificação e *debug* (i.e., não considere o facto da simulação pós-síntese ser mais demorada do que a simulação funcional).

[2.5 valores]

3 - A implementação de um sinal de *reset* é fundamental para permitir definir um estado inicial coerente em todos os registos de um sistema. Se o sinal de *reset* for usado diretamente de uma entrada primária do circuito (ver exemplo abaixo) e se for ativado de forma assíncrona com o sinal de relógio (por exemplo, através de um simples botão de pressão atuado manualmente) poderá conduzir a uma inicialização incorreta dos registos que controla.

```
module toplevel( input reset, input clock, ... );
...
always @(posedge clock or posedge reset)
if (reset)
// initialization actions
else
// non-reset behavior
```

[1 valor]

a) Explique por que razão pode ocorrer essa falha.

[1.5 valores]

b) Mostre como resolver esse problema, considerando um sistema síncrono com um único sinal de relógio e com um sinal de *reset* assíncrono, tal como mostrado no exemplo acima.

[4 valores]

4 - Considere o excerto de código Verilog seguinte em que as memórias M2 e M3 contêm o mesmo conjunto de dados:

```
// signals 'addr[5:0]' and 'enable' are inputs
reg [ 7:0] M1[0:63];
reg [19:0] M2[0:255], M3[0:255];
reg [19:0] p1, p2;
reg [ 7:0] i, j, a2, a3;
reg [19:0] yout;

always @(posedge clock)
if (reset)
  // initialize all register with zeros
else
begin
  yout <= enable ? ( p1 + p2 ) >> 1 : yout;
  a2   <= j + i;
  a3   <= j + i + 1;
  p1   <= M2[ a2 ];
  p2   <= M3[ a3 ];
  i    <= enable ? (i + 2) : 8'd0;
  j    <= M1[ addr ];
end
```

[2 valores]

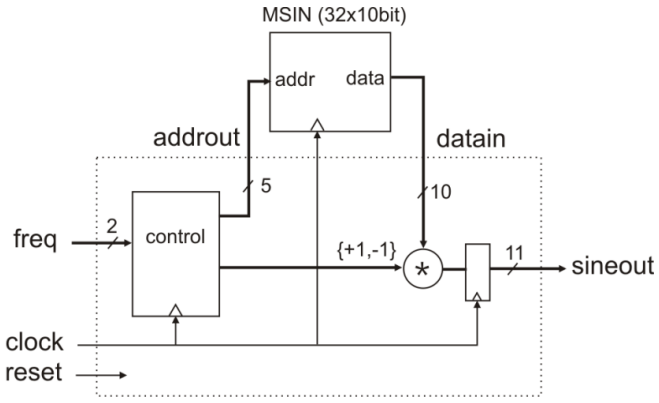
a) Desenhe um diagrama de blocos que mostre o circuito digital representado por este modelo. Considere que os vetores M1[ ], M2[ ] e M3[ ] são implementados como memórias com leitura assíncrona e represente-os como tal. Para simplificar o desenho não represente as partes do circuito que implementam a ação de *reset*.

[2 valores]

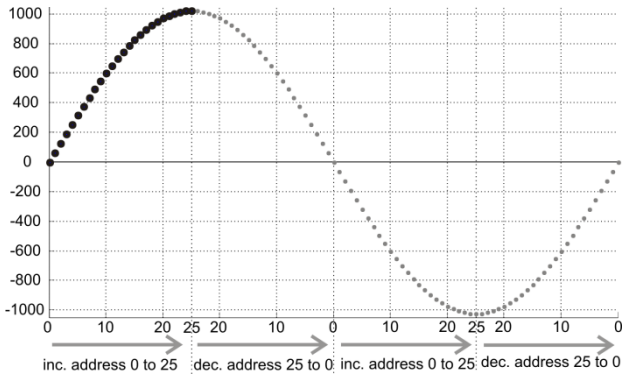
b) Numa reunião da equipa responsável pelo desenho deste bloco, Mr. X defendeu que seria possível redesenhar este circuito gastando menos área e ficando mais rápido, porque se sabe que o período de relógio que vai ser usado é cerca de 10X superior ao tempo de propagação de um somador de 32 bits e cerca de 8X superior ao tempo de propagação das memórias M2 e M3. Explique como referindo-se às eventuais implicações que isso poderá ter relativamente ao resto do sistema que faz uso deste bloco.

[4 valores]

5 - Pretende-se desenvolver um módulo digital capaz de gerar sinais sinusoidais com 11 bits por amostra (em complemento para 2) e com frequência selecionável entre 100 kHz, 50 kHz, 25 kHz e 12.5 kHz. O diagrama de blocos da figura representa o circuito a implementar.



Uma memória só de leitura, externa ao circuito a desenvolver (**MSIN**), contém uma tabela com 26 amostras que representam os valores da função *sin(x)* no primeiro quadrante multiplicada por 1023 (para valores de x no intervalo $[0,\pi/2]$ - pontos representados a negro na figura abaixo). Tirando partido da simetria da função *sin(x)* e endereçando essa memória de forma adequada é possível obter uma sequência de valores que representam a função *sin(x)* para todo o seu período. Para gerar a função sinusoidal de frequência 100 kHz, com uma amostra por período de relógio a sequência de endereços a aplicar à memória terá de ser 0, 1, 2, ..., 24, 25, 24,..., 2, 1, 0, 1, 2, ..., trocando o sinal do valor lido da memória durante o 3º e 4º varrimento da memória (3º e 4º quadrantes).



Usando um sinal de relógio de 10 MHz e incrementando/decrementando os endereços da memória de 1 em 1 obtém-se uma sinusoide com uma frequência de 100 kHz. Se o endereço aplicado à memória for incrementado/decrementado com um passo de 0.5 (ou incrementado/decrementado a cada 2 ciclos de relógio) a sinusoide gerada terá uma frequência de 50 kHz (para 25 kHz é usado um passo de 0.25 e para 12.5 kHz um passo de 0.125).

Construa um modelo em Verilog HDL que implemente este circuito, seguindo as restrições de codificação para que seja sintetizável e adotando como critério de otimização a minimização da área do circuito. A memória **MSIN** (32 palavras de 10 bits) que contém as 26 amostras da função sin(x) será externa a este módulo e seu acesso de leitura é síncrono com o flanco ativo do sinal de relógio. O endereço dessa memória é a saída **addrout** e o dado lido da memória é ligado à entrada **datain**. O sinal de reset deverá ser síncrono com o sinal de relógio e ativo com o nível lógico alto. A entrada **freq** define a frequência do sinal de saída, conforme a tabela seguinte:

Entrada **freq**	2'b00	2'b01	2'b10	2'b11
Frequência gerada	100 kHz	50 kHz	25 kHz	12.5 kHz

```
module sinegen( input clock,
                input reset,
                input  [ 1:0] freq,
                output [ 4:0] addrout,
                input  [ 9:0] datain,
                output [10:0] sineout
              )


endmodule;
```

[4.5 valores]

6 - No projeto de um sistema de processamento de vídeo pretende-se construir um sistema digital para calcular em tempo real a convolução 2D (janela de 5 x 5 pixel) sobre um sinal de vídeo digital, com uma latência mínima. Durante a receção de uma imagem completa do vídeo (*frame*) a imagem é recebida linha a linha (de cima para baixo da imagem), e os pixel de cada linha são recebidos um a um (da esquerda para a direita) à frequência do sinal de relógio de pixel igual a 112 MHz (*pixel clock*). Dois sinais de sincronismo adicionais, não relevantes para o sistema a desenvolver, sinalizam o fim/início de cada linha (**HSYNC**) e o fim/início de cada imagem completa (**VSYNC**). Entre a chegada do último pixel de uma imagem e o início do 1º pixel da imagem seguinte ocorrem pelo menos 256 ciclos do relógio de pixel.

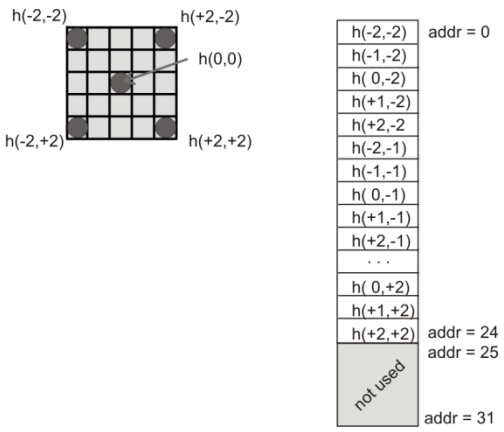
A convolução 2D entre uma imagem digital representada pela matrix *Pixelin(x,y)* e a resposta impulsional (ou *kernel*) de 5x5 *h(i,j)*, produz o pixel de saída nas coordenadas *(x,y), Pixelout(x,y)* calculado como:

$$Pixelout(x,y) = \sum_{i=-2}^{+2} \sum_{j=-2}^{+2} Pixelin(x+i, y+j) \times h(i,j)$$

O sistema a implementar faz uso de uma estrutura de memória dedicada, baseada em registos de deslocamento, que permite em cada ciclo do relógio de pixel (*pixel clock*) ter acesso em paralelo aos 25 pixels vizinhos (*Pixelin(x+i, y+j)*) do pixel a ser calculado (*Pixelout(x,y)*). Os pixel a processar (*Pixelin(x+i, y+j)*) e o resultado (*Pixelout(x, y)*) são valores de 8 bits sem sinal.

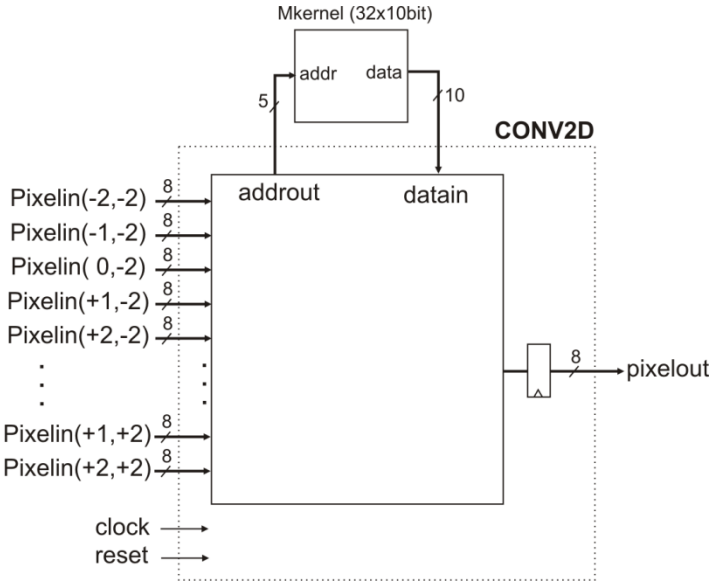
O *kernel* *h(i,j)* é composto por 25 valores de 10 bits com sinal, representando 3 bits a parte inteira e 7 bits a parte fracionária. Estes valores estão armazenados numa memória (**Mkernel**) de 32 x 10 bits a partir do endereço 0 e organizados na memória da forma mostrada na figura abaixo. Esta memória tem acesso de leitura assíncrono e pode ser escrita por um sistema de controlo externo mas apenas durante o período de intervalo entre frames (com

a duração de 256 ciclos do relógio de pixel). Tal como foi considerado no projeto laboratorial, assume-se que os valores *h(i,j)* são tais que o resultado da convolução 2D dá sempre um resultado positivo e inferior a 256 (representável por isso em 8 bits).



Sabe-se que, na tecnologia que vai ser usada na implementação deste sistema, um multiplicador de 8 x 10 bits (resultado de 18 bits) apresenta um tempo de propagação de 4 ns e um somador de 20 bits tem um tempo de propagação de 1.5 ns (estes valores são naturalmente aproximados).

A figura ao lado mostra um diagrama de blocos simplificado, em que o bloco **CONV2D** representa o módulo a desenvolver. As entradas **Pixelin(x,y)** contêm os 25 pixels da região da imagem de entrada na vizinhança do pixel a calcular, que devem ser multiplicados pelos coeficientes *h(i,j)*. Estas 25 entradas são atualizadas a cada ciclo do relógio de pixel.



[1.5 valores]

a) Mostre como implementar o módulo **CONV2D** usando um sinal de relógio com frequência igual ao do relógio de pixel (112 MHz) e minimizando a latência do circuito. Descreva e justifique devidamente a solução apresentada, ilustrando-a com diagramas de blocos ou com excertos de código Verilog. Não considere os casos particulares que ocorrem quando o pixel a calcular está na periferia da imagem e não existem alguns dos pixel vizinhos a processar (por exemplo para *(x,y) = (0,0)*)

[1.5 valores]

b) Com o objetivo de reduzir a área do circuito, foi proposto usar para este bloco um sinal de relógio com frequência dupla ou tripla da do relógio de pixel (224 MHz ou 336 MHz, com períodos iguais a 4.46 ns e 2.98 ns, respetivamente). Mostre como alterar o circuito proposto na alínea anterior de forma a tirar partido de um sinal de relógio mais rápido para obter um circuito mais pequeno. Indique uma medida aproximada da redução percentual de área que se obtém com a solução proposta. Ignore as questões relacionadas com a necessidade de se garantir o sincronismo entre os dois sinais diferentes de relógio.

[1.5 valores]

c) O mesmo sistema poderá ser usado para realizar convoluções 2D com *kernels* 3x3 em vez de 5x5, bastando para isso que os 16 coeficientes da periferia da matriz 5x5 (*h(-2,…), h(+2,…), h(…,-2) e h(…,+2)* sejam iguais a zero (ou considerados como zero). Admita que o sistema de controlo que carrega aquela memória fornece ao seu sistema um bit que indica se o *kernel* carregado na memória deve ser usado como 3x3 ou 5x5. Mostre como poderia usar essa informação para modificar o seu sistema de forma a conseguir reduzir o consumo dinâmico de energia quando for selecionado o kernel 3x3 e indique uma estimativa percentual aproximada dessa redução, referindo-se apenas ao consumo dinâmico do bloco **CONV2D**.

..:: FIM ::..