FEUP
Universidade do Porto
Faculdade de Engenharia

Departamento de Engenharia Eletrotécnica e de Computadores

EEC0055 - Projeto de Sistemas Digitais

4º ano - 1º semestre

Correção do exame de 23 de janeiro de 2019

[4 valores]

1 - Ao longo do projeto de um sistema digital para tecnologias digitais integradas, semelhantes à que foi usada nos trabalhos laboratoriais, são realizadas duas transformações fundamentais (recorrendo a ferramentas de projeto) em que o modelo do sistema é traduzido do código fonte construído pelo projetista (p.ex. Verilog) até uma descrição que representa a sua organização ao nível físico. Explique:

[2 valores]

a) Qual é a primeira dessas transformações, que tem por origem o código fonte Verilog, referindo-se ao resultado que é produzido e às razões pelas quais a execução desta tarefa pode obrigar ao re-desenho do código fonte RTL.

A primeira transformação é a síntese RTL (ou síntese lógica), que traduz o código Verilog comportamental e/ou estrutural num modelo de um circuito lógico formado pela interligação de componentes lógicos elementares, disponíveis como blocos primitivos da tecnologia alvo. Após a realização da síntese RTL, pode ser necessário redesenhar o código fonte (Verilog) se:

(i) o circuito criado não satisfizer requisitos não funcionais definidos para o projeto, como a frequência estimada para o sinal de relógio ou a área (ou quantidade de recursos lógicos) usados para a sua implementação;

(ii) os modelos comportamentais usados como origem para este processo não respeitarem as regras de modelização consideradas pela ferramenta de síntese, o que pode conduzir à criação de circuitos cujo comportamento, após a posterior implementação física, não é coerente com o comportamento do modelo funcional.

[2 valores]

b) Qual é a segunda dessas tarefas que produz a representação do circuito ao nível físico, referindo-se também aos motivos que podem obrigar ao re-desenho do código fonte RTL, ou à re-execução da tarefa anterior, depois de realizar esta transformação.

A transformação referida consiste na definição da localização física de cada componente lógico elementar que forma o circuito criado pelo processo de síntese RTL e da construção das interligações entre esses componentes. Esta tarefa designa-se habitualmente por "Place & Route" e um dos principais motivos que pode obrigar a re-executar a síntese RTL (com diferentes estratégias de otimização) ou mesmo re-desenhar o código comportamental é a não satisfação de requisitos temporais associados aos sinais lógicos, tal como a frequência pretendida para o sinal de relógio, ou atrasos de propagação máximos/mínimos ao longo de certos caminhos (por exemplo, de uma entrada para um registo ou de um registo para uma saída).

[4 valores]

2 – Considere as ferramentas de projeto e de simulação que foram usadas nos trabalhos laboratoriais e um cenário semelhante ao do 3º projeto laboratorial, em que teve de desenhar um bloco que foi mais tarde integrado num outro projeto.

[2 valores]

a) Imagine que o processo de verificação funcional do código Verilog que implementa o seu bloco foi completado com sucesso (antes deste ser integrado no sistema). No entanto, depois da sua integração no projeto final e após completar a implementação verificou-se o sistema não cumpre a função esperada e que a falha se deve ao seu bloco e/ou à forma como foi integrado no resto do sistema. Apresente exemplos de motivos que podem explicar essa situação (pelo menos 2) e diga como poderiam ser identificados os problemas que refere.

Uma falha detetada após a integração de um bloco (admitindo que esse bloco está corretamente modelizado e foi devidamente verificado funcionalmente) pode resultar de:

(i) Não satisfação de requisitos temporais, devido ao facto de ao efetuar a ligação entre esse bloco e outros, poderem ser criados caminhos combinacionais cujo atraso de propagação excede o máximo admitido;

(ii) Não se implementar corretamente os mecanismos de sincronização que são fundamentais para interligar sinais provenientes de (ou destinados a) circuitos a operar com domínios diferentes de relógio;

(iii) Troca de portos de interface ao instanciar o bloco. Apesar de ser emitida uma mensagem de aviso quando o número de bits de um sinal ligado a um porto de interface não coincide com o número de

bits desse porto, se dois sinais com o mesmo número de bits forem trocados entre si, isso não produz qualquer aviso e é um erro que naturalmente compromete o projeto (imagine que ao instanciar um bloco é trocado o clock com o reset, ou o reset com um clock enable!). Esta situação pode acontece quando a ligação de sinais a porto é feita por posição, razão pela qual é recomendável que isso seja feito especificando os nomes dos portos que devem ser ligados a cada sinal.

Os dois primeiros erros (e, em geral, qualquer falha relacionada com a temporização dos sinais lógicos) serão detetados apenas após a implementação física (Place&Route), por análise dos relatórios de análise temporal, ou por simulação "post-route", enquanto que o erro (iii) poderia ser facilmente detetado com a realização de uma simulação funcional.

[2 valores]

b) Uma das especificações que lhe foram definidas para o bloco que projetou foi que o sistema deveria ser síncrono com um único sinal de relógio de 150 MHz. Diga, justificando, em que fase do processo de projeto é possível verificar se esse requisito é ou não satisfeito.

Só após a síntese física do circuito (ou Place&Route) é possível obter os modelos temporais precisos que caracterizam os tempos de propagação dos sinais ao longo do circuito. Esses tempos incluem os atrasos de propagação associados aos elementos lógicos e às suas interligações e só com essa informação é possível estimar o período mínimo do sinal de relógio (ou a sua frequência máxima) como sendo o tempo máximo de propagação do caminho mais longo entre registos atuados por esse sinal de relógio.

[4 valores]

3 - A construção de modelos Verilog para alimentar processos de síntese RTL deve seguir um conjunto de regras de codificação de circuitos combinacionais e sequenciais síncronos, que são universalmente aceites pelas ferramentas de síntese. Os dois excertos de código Verilog seguintes, que pretendem representar um circuito sequencial e outro combinacional, violam regras elementares de codificação (admita que os sinais referidos estão devidamente declarados):

(i)

```
always @*
begin
  if ( clken )
  begin
    if ( ~muxsel )
      zsum = ra + rb[12:0];
    else
      zdiff = rb - ra[12:0];
  end
  else
  begin
    zsum = 0; zdiff = 0;
  end
end
```

(ii)

```
always @(posedge clock or posedge rst)
begin
  if ( rst )
    zout = output1;

  ztemp = Ra * Ra - Rb;
  zout  = ztemp * 1.34;
end
```

[2 valores]

a) Diga justificando quais os erros que identifica no código que pretende modelizar um circuito combinacional e explique como o deveria corrigir.

*O excerto de código que pretende representar um circuito combinacional é o (i) (**always @* …**). O erro é não estar definido o valor do sinal **zsum** para a condição **clken == 1** e **mulsel == 1** e de **zdiff** para a condição **clken == 1** e **mulsel == 0**. Este erro de codificação (note que não é um erro de Verilog!) faz com que o circuito sintetizado não seja combinacional mas sim sequencial assíncrono (i.e., sem ser sincronizado por uma sinal de relógio), implementado com elementos de memória do tipo "latch" transparente. A correção desse erro obriga a especificar o valor lógico que esses sinais devem assumir nessas condições, o que será dependente da aplicação. Admitindo que o valor zero seria possível, o código corrigido poderia ser:*

```
always @*
begin
  if ( clken )
  begin
    if ( ~muxsel )
      zsum = ra + rb[12:0]; zdiff = 0;
    else
      zdiff = rb - ra[12:0]; zsum = 0;
  end
  else
  begin
    zsum = 0; zdiff = 0;
  end
end
```

[2 valores]

b) Diga justificando quais são os erros no código que pretende modelizar um <u>circuito sequencial</u> e diga como corrigir.

*O excerto de código que pretende representar um circuito sequencial é o (ii) (**`always @(posedge clock …)`**. Este excerto de código apresenta 3 erros:*

*(i) A ação de reset assíncrono está mal construída, faltando o "**`else`**" na condição que avalia o sinal de reset (**`rst`**). Note que não é incorreto usar um sinal de reset assíncrono e a forma de o construir na lista de eventos da declaração **`always`** é como está apresentado;*

(ii) As atribuições dentro de um bloco síncrono com sinal de relógio devem ser todas do tipo "non-blocking", caso contrário a simulação desse código pode não traduzir o mesmo comportamento do circuito que é sintetizado.

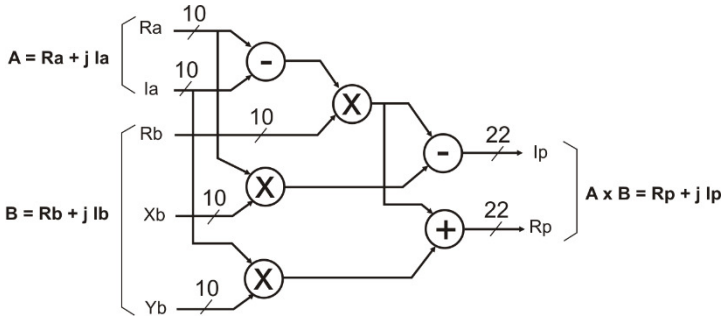
(iii) É ilegal o uso da constante não inteira 1.34, já que apenas são sintetizáveis expressões que contenham constantes inteiras. O produto por 1.34 poderia ser aproximado pelo produto inteiro pela constante 1372 (≈ 1.34 x 1024), seguida da divisão por 1024 ou deslocamento de 10 bits para a direita.

O código corrigido pode ser:

```
always @(posedge clock or posedge rst)
begin
  if ( rst )
    zout <= output1;
  else
  begin
    ztemp <= Ra * Ra – Rb;
    zout  <= ( ztemp * 1372 ) >>> 10;
  end
end
```

[4 valores]

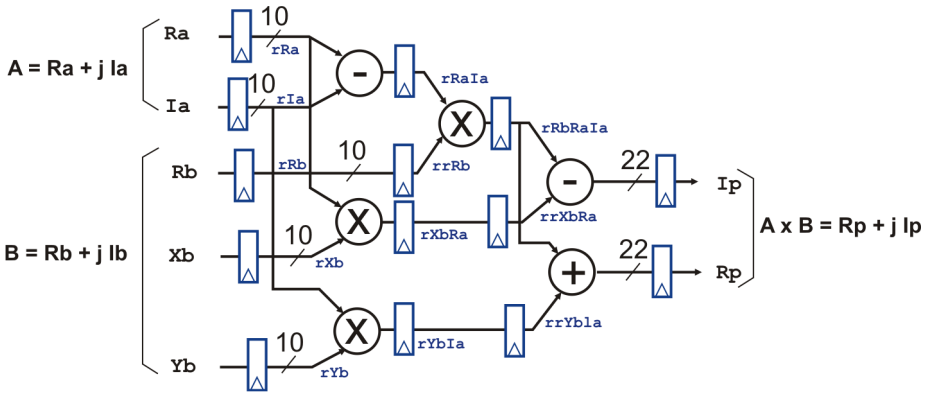
4 - O diagrama da figura mostra uma implementação do produto de 2 números complexos **A** e **B** dados na forma R + jI. O operando **B** = Rb + jIb é representado pelas 3 componentes **Rb**, **Xb** = Rb+Ib e **Yb** = Rb-Ib, o que permite implementar o produto complexo com apenas 3 multiplicadores em vez dos 4 que seriam necessários usando o cálculo direto de (Ra+jIa) x (Rb + jIb) = (Ra x Rb – Ia x Ib) + j(Ia x Rb + Ra x Ib). Os 5 operandos de entrada são inteiros com sinal de 10 bits e os resultados **Rp** e **Ip** (parte real e parte imaginária) são também inteiros com sinal de 22 bits.



[2.5 valores]

a) <u>Construa um módulo Verilog (sintetizável)</u> que realize o cálculo representado pelo diagrama da figura, onde os operadores aritméticos são implementados como circuitos combinacionais. <u>Devem ser criados 5 registos</u> nas entradas, para os 5 operandos, e <u>2 registos para os dois resultados</u>. O <u>*datapath* deve ser *pipelined*</u> de forma a que <u>a frequência máxima de relógio seja condicionada pelo tempo máximo de propagação dos multiplicadores</u>.

Se a frequência máxima de relógio deve ser condicionada pelo atraso de propagação dos multiplicadores, então o circuito a construir deve ter registos nas entradas e saída dos multiplicadores. Isso resultará num circuito com dois estágios de pipeline, para além dos registos de entrada e saída, cujo diagrama de blocos é o seguinte (os nomes dos sinais referem-se aos nomes usados no modelo Verilog):



O código Verilog que implementa esse circuito é o seguinte, onde, para simplificar, não se representa a declaração do módulo e se consideram os dados como "unsigned":

```
// Input registers:
reg [ 9:0] rRa, rIa, rRb, rXb, rYb;

// First pipeline stage:
reg [10:0] rRaIa;
reg [ 9:0] rrRb;
reg [19:0] rXbRa, rYbIa;

// Second pipeline stage:
reg [20:0] rRbRaIa;
reg [19:0] rrXbRa, rrYbIa;

// Output registers:
reg [21:0] Ip, Rp;

always  @(posedge clock)
if ( reset )
  // set all registers to 0;
else
begin
  // Load input registers;
  rRa <= Ra; rIa <= Ia; rRb <= Rb; rXb <= Xb; rYb <= Yb;

  // First pipeline stage:
  rRaIa <= rRa - rIa;  rrRb <= rRb;  rXbRa <= rRa * rXb;  rYbIa <= rIa * rYb;

  // Second pipeline stage:
  rRbRaIa <= rRaIa * rrRb; rrXbRa <= rXbRa; rrYbIa <= rYbIa;

  // Output registers:
  Ip <= rRbRaIa - rrXbRa;  Rp <= rRbRaIa + rrYbIa;
end
```

[1.5 valores]

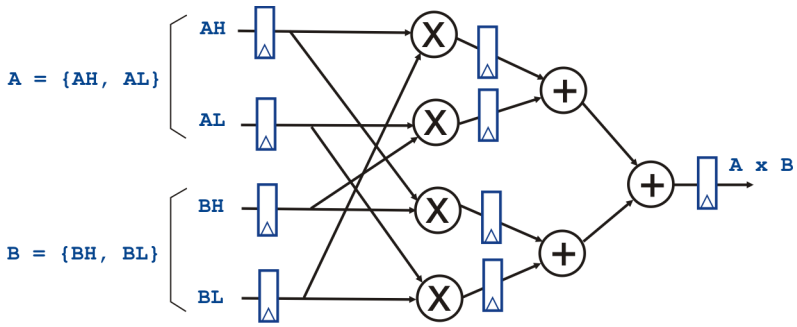
b) Depois de implementar esse circuito verificou-se que a frequência máxima de relógio não ultrapassava os 160 MHz (período de 6.3 ns), quando o circuito necessita de operar com uma frequência mínima de relógio de 200 MHz (período 5 ns). Para resolver isso será necessário acrescentar estágios de *pipeline* adicionais de forma a dividir o circuito lógico combinacional que implementa o multiplicador. Explique como implementaria essa solução recorrendo a diagramas ou código Verilog para suportar a sua explicação.

A indicação que o período mínimo de relógio é igual a 6.3 ns indica que esse será, aproximadamente, o tempo de propagação do multiplicador combinacional, que é o bloco entre registos com maior atraso. Com um único andar extra de pipeline, inserido a "meio" do circuito combinacional que implementa o multiplicador, aquele tempo de propagação será (idealmente!) dividido a meio e assim poderá ser possível reduzir o período de relógio mínimo para menos dos 5 ns pretendidos.

Uma forma possível de implementar isso, embora não dividindo o tempo de propagação perfeitamente em duas partes iguais, consiste em realizar o produto de dois números de 10 bits implementado a expressão seguinte, onde AL e BL representam os 5 bits menos significativos dos operandos A e B, e AH e BH os respetivos 5 bits mais significativos:

```
A x B = {AH, AL} x {BH, BL} = ( AL x BL ) + ( AL x BH + BL x AH ) << 5 + ( AH x BH ) << 10;
```

O diagrama de blocos mostra uma implementação pipelined desta expressão, onde não estão explicitamente representados os deslocamentos de bits necessários para alinhar os produtos parciais:

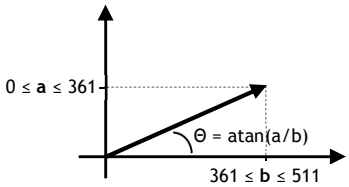


Como os produtos seriam agora de operandos de 5 bits em vez de 10, é de estimar que o tempo de propagação de cada multiplicador seja reduzido para aproximadamente metade dos de 10 bits, e a árvore de somadores terá um tempo de propagação inferior ao dos multiplicadores. Assim, um único nível de pipeline entre os multiplicadores e a árvore de somadores, seria, em princípio, suficiente para garantir aquela frequência de relógio.

[4 valores]

5 - No projeto dum sistema digital para processar os sinais de um sensor de posição angular é necessário implementar um calculador da função **atan(a/b)**, apenas para o primeiro octante. O sistema em que esse projeto se insere destina-se a ser implementado como um circuito integrado de aplicação específica (ASIC) e é objetivo global do projeto a minimização da sua área.

Os operandos **a** e **b** são inteiros positivos de 9 bits e representam, respetivamente, as componentes segundo Y e X de um vetor cujo ângulo se pretende calcular e que tem módulo aproximadamente igual a 511. Assim, os valores de **a** e **b** são condicionados pelas relações seguintes: **0 ≤ a ≤ 361** e **361 ≤ b ≤ 511**:



O bloco a desenvolver deverá ser capaz de processar dados apresentados continuamente ao ritmo de 100 kHz e poderá operar com um sinal de relógio com frequência máxima de 2 MHz. O resultado produzido deverá ser um inteiro de 9 bits representando o ângulo **atan(a/b)** em unidades de décima de grau (entre 0 e 450). O atraso entre a apresentação dos operandos **a** e **b** e a geração do resultado não pode exceder 20μs (dois períodos de 100 kHz ou 40 períodos do relógio de 2 MHz).

Numa primeira reunião da equipa de projeto, 3 colegas propuseram implementações alternativas para este módulo que apresentaram da seguinte forma:

1) *A minha proposta é muito simples e escreve-se em apenas 2 linhas de código: primeiro calcula-se o quociente **a/b** com **assign x = a / b**; e depois obtém-se o valor da função atan() recorrendo a uma tabela, escrevendo **assign atan = ROM_ATAN[x]**; onde **ROM_ATAN[ ]** é uma tabela implementada como uma memória pré-carregada com os valores de atan(x).*

2) *Concordo com essa ideia geral [calcular primeiro o quociente **a/b** e usar depois uma tabela], mas como é possível usar até 40 ciclos de relógio para calcular o resultado tenho uma solução melhor que permite reduzir substancialmente a área/complexidade do circuito.*

3) *Não concordo com as vossas propostas. Há uma outra solução ainda mais simples do que de 2), que também faz uso de alguns dos 40 ciclos de relógio disponíveis entre cada dois conjunto de de operandos e que só necessita de uma memória pequena, com menos de 16 palavras.*

[1.5 valores]

a) Coloque-se no papel do projetista 2). Explique em que consistirá a solução que ele apresenta e diga porque razão é mais simples da que foi proposta por 1).

A solução proposta por 2) baseia-se em usar os 40 ciclos de relógio disponíveis entre cada cálculo para realizar operações com circuitos sequenciais iterativos, em vez de circuitos puramente combinacionais como é sugerido pela opção apresentada pelo projetista 1). A mesma abordagem foi seguida no projeto laboratorial, onde se fez uso de multiplicadores sequenciais em vez de combinacionais para realizar as 5 multiplicações (e houve mesmo quem usasse um único multiplicador para realizar as 5 operações). Assim, pode-se concluir que o projetista 2) sugere usar um divisor sequencial, já que o número de ciclos de relógio disponíveis (40) é largamente superior ao número de ciclos de relógio necessários para um divisor sequencial realizar a divisão de 2 números de 10 bits.

[1.5 valores]

b) Assuma agora o papel do projetista 3). Diga, justificando, qual será a solução a que se refere.

Uma solução mais simples para realizar o cálculo de atan(a/b) e que não requer o cálculo explícito de a/b consiste em usar o algoritmo CORDIC, implementado de forma iterativa. O algoritmo realizará o cálculo em 10 + 3 ou 4 ciclos de relógio (dependendo de se implementar ou não processos de arredondamento) e necessita apenas de dois somadores/subtratores, registos e uma tabela com os valores de atan(2^-i) com um número de entradas igual os número máximo de iterações, que neste caso será seguramente inferior a 16.

[1 valor]

c) Com o objetivo de procurar reduzir o consumo de energia, a frequência máxima de relógio foi redefinida para apenas 100 kHz e por isso o circuito terá de produzir um resultado em no máximo 2 ciclos de relógio. Explique se nestas condições poderá ser atingido o objetivo de reduzir o consumo de energia (apenas relativa a esse bloco), tendo em atenção que neste novo cenário as soluções propostas por 2) e 3) já não podem ser adotadas (mas note que isso não significa que tenha de ser seguida a solução proposta pelo 1) !)

A potência consumida por um circuito digital tem duas parcelas principais:

(i) *uma que depende da atividade de comutação do circuito (potência dinâmica) e que é diretamente proporcional à frequência de operação do circuito, dependendo também do número de sinais lógicos que comutam e das características físicas das suas ligações elétricas. Embora a frequência de relógio tenha sido reduzida em 20x, como agora o circuito digital terá de ser necessariamente maior, possivelmente usando mesmo um divisor combinacional, a redução na potência consumida será necessariamente menor do que 20x;*

(ii) *a outra parcela importante é devida às correntes de fugas dos circuitos e pode-se estimar que esta componente seja proporcional à dimensão do circuito digital.*

O saldo final da potência consumida dependerá de variados fatores, entre os quais o peso do consumo devido às correntes de fugas versus o consumo dinâmico e também a arquitetura adotada para implementar a função lógica. Por exemplo, se na tecnologia em causa o consumo estático fosse responsável por 50% do consumo global e o consumo dinâmico pelos restantes 50%, reduzir 10x o consumo dinâmico e aumentar o consumo estático em apenas 2x conduziria a um aumento de 5% no consumo global.

-◖= Fim =◗-