FEUP Universidade do Porto Faculdade de Engenharia

Departamento de Engenharia Eletrotécnica e de Computadores

Projeto de Sistemas Digitais

4º ano - 1º semestre
Exame – 23 janeiro 2014 – CORREÇÃO V1.0 (27 janeiro 2014)
(comentários adicionais a verde)

[5 valores]

1 - Durante o desenvolvimento do último projeto laboratorial optou-se por não realizar a fase de verificação pós-síntese nem pós-layout do projeto completo. Essa opção resultou essencialmente de falta de tempo e da necessidade de para isso se ter de desenvolver um modelo de verificação complexo e que resultaria num tempo muito elevado de simulação (a mesma abordagem foi seguida no projeto desenvolvido com a câmara digital).

[2,5 valores]

a) Imagine que verificou com sucesso a funcionalidade do módulo que desenvolveu, simulando o código RTL que escreveu. No entanto, depois de o integrar no restante projeto dado e o implementar concluiu que o circuito final não funcionava como esperado. Enuncie 3 possíveis causas para isso e diga em que simulação posterior à verificação funcional (pós-síntese ou pós-layout) poderiam ser detetadas e identificadas.

Erro na codificação em Verilog que, apesar de permitir sintetizar um circuito lógico pode fazer com que o simulador interprete um comportamento diferente do circuito que é sintetizado; isto pode acontecer quando são usados atrasos explícitos em sinais, codificados processos síncronos com clock (`always @(posedge clock)`*) que usem atribuições do tipo* blocking *ou quando são codificados circuitos combinacionais que infiram blocos sequenciais do tipo latch (*`if-then-else` *incompletos). Nos dois primeiros casos o erro é detetável na simulação pós-síntese, mas no segundo só na simulação pós-layout, já que o comportamento das* latches *que são criadas irá depender dos tempos de propagação, só conhecidos após a implementação física (*place&route*).*

Erro nas ligações ao integrar o bloco desenvolvido no projeto completo, como por exemplo não definir corretamente o número de bits dos barramentos que os interligam ou esquecimento de ligar uma saída: detetável na fase de verificação pós-síntese

Erro por não satisfação dos requisitos temporais definidos para a implementação, o que na sua forma mais simples determina um valor mínimo para o período de relógio: detetável apenas na fase de verificação pós-layout e só se a simulação realizada ativar o caminho lógico que excede essa restrição.

[2,5 valores]

b) Considere agora que tinha implementado o seu projeto com uma restrição temporal que estabelecia 150 MHz como frequência pretendida para o sinal de relógio. Após a fase de *place&route* foi reportado pelas ferramentas de análise temporal uma frequência máxima de apenas 100MHz. Ainda assim implementou o circuito na FPGA e verificou que funcionava corretamente com o sinal de relógio de 150 MHz. Diga quais as verificações adicionais que teria de fazer para poder garantir que o circuito seria capaz de operar àquela frequência, dentro das condições normais de trabalho para o dispositivo utilizado (tensão de alimentação e temperatura).

A frequência máxima calculada para o sinal de relógio pelas ferramentas de implementação é determinada pela análise temporal estática do circuito implementado e pelo cálculo dos tempos de propagação de todos os caminhos desde um flip-flop (sinal de relógio) até à entrada D do seguinte, passando pelo circuito combinacional que define o estado seguinte do flip-flop a jusante. Isto pressupõe que <u>todos</u> os flip-flops ativados por um certo sinal de relógio capturam os sinais nas suas entradas em <u>todas</u> as transições de relógio, o que não acontece em muitos casos. Por exemplo, se o caminho crítico (grosso modo, o circuito combinacional que apresenta o maior tempo de propagação do circuito combinacional entre registos) que limita a frequência do sinal de relógio tiver os flip-flops a jusante que nunca são carregados na transição de relógio a seguir ao carregamento do primeiro, essa não limitação não é verdadeira.

Como as ferramentas de análise temporal estática não consideram essas situações (até porque

podem depender dos dados de entrada), consideram sempre o caso pior que pode nunca acontecer.

Para completar essa verificação será necessário inspecionar os relatórios de análise temporal para verificar que os caminhos que são reportados como limitando a frequência de relógio nunca são excitados à frequência de relógio (pelos motivos referidos acima) ou recorrer a uma simulação pós-place&route, com um conjunto de estímulos adequados para ativar as secções do circuito indicadas como críticas.

[4 valores]

2 - Os dois excertos de Verilog seguintes são funcionalmente equivalentes, representando a mesma máquina de estados finitos (modelo de Moore). No entanto, os circuitos lógicos que resultam da sua síntese RTL são estruturalmente diferentes. Admita que os sinais **`St[1:0]`**, **`Y`**, **`I1[15:0]`** e **`I2[15:0]`** estão devidamente declarados.

```
always @(posedge clock)
if ( reset )
begin
  St <= INI;
  Y <= 1'b0;
end
else
  case ( St )
    INI: if ( I1 == 0 )
         begin
           St <= RUN;
           Y <= 1'b1;
         end
    RUN: if ( I1 > I2 )
         begin
           St <= WAIT;
           Y <= 1'b0;
         end
    WAIT: St <= INI;
  endcase
```

Solução A

```
always @(posedge clock)
if ( reset )
begin
  St <= INI;
end
else
  case ( St )
    INI: if ( I1 == 0 )
         begin
           St <= RUN;
         end
    RUN: if ( I1 > I2 )
         begin
           St <= WAIT;
         end
    WAIT: St <= INI;
  endcase

assign Y = ( St == RUN );
```

Solução B

[2 valores]

a) Diga em que diferem esses dois circuitos (os circuitos lógicos que os modelos representam, não o código Verilog dado!).

*Na solução **A** a saída Y é registada e por isso existirá um flip-flop que a cada mudança de estado será carregado com o valor de Y para cada estado (como é um modelo de Moore, a saída não depende diretamente das entradas). Na solução **B** a saída Y é a saída de um circuito combinacional.*

[2 valores]

b) Ambas as soluções podem ser justificadas em função de diferentes requisitos de projeto. Explique em que cenários optaria pela solução A e quando deveria ser utilizada a codificação exemplificada em B.

*O facto de no modelo **A** saída Y ser registada pode fazer com que seja sintetizado um flip-flop adicional para implementar Y, tornando-o por isso num circuito (potencialmente - ver nota abaixo) mais complexo do que a solução **B** em que Y é gerado por um circuito combinacional. Apesar de ser apenas um flip-flop adicional, a solução **B** poderia por isso ser vantajosa em situações em que seja crítico o espaço ocupado.*

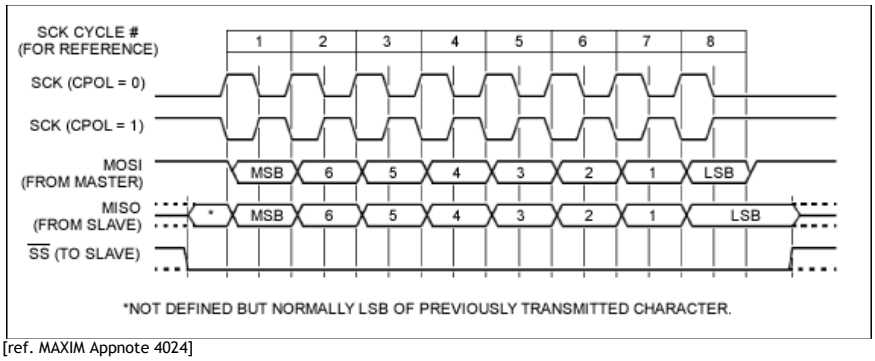
*No entanto, o facto de na solução **B** a saída Y ser produzida por um circuito combinacional faz com que durante as mudanças de estado em Y possam ocorrer transições supérfluas (*glitches*). Essas transições podem ser críticas se esse sinal for usado como sinal de relógio de outro circuito a jusante e também podem conduzir a gastos de energia adicionais pelo facto dessas transições se propagarem pelos circuitos que alimentam. Neste caso a solução **A** seria preferível.*

*Apesar de na solução **A** a saída Y ser uma saída registada, a otimização realizada durante a síntese RTL pode inferir que o flip-flop que realiza a saída Y é logicamente a outro flip-flop e implementar apenas um deles (isto pode ser ajudado com uma escolha adequada dos códigos de estado). Por exemplo, se a codificação de estados for INI=2'b00, RUN=2'b10 e WAIT=2'b01 então a saída Y pode ser implementada como o bit mais significativo do registo de estado, sem ser necessário um flip-flop adicional.*

[6 valores]

3 - Um interface SPI é um interface série síncrono unidirecional com 3 fios (4 fios para uma ligação

bidirecional), usado em diversos dispositivos integrados, sendo atualmente suportado por vários micro-processadores e micro-controladores. A figura mostra um diagrama temporal que ilustra a sequência de sinais para uma possível configuração de um interface SPI bidirecional de 8 *bits*, onde o parâmetro CPOL configura a polaridade que é usada para o sinal de relógio de dados. O interface é controlado por um dispositivo "*master*" que produz o sinal /SS (*SPI Select*) e o sinal SCK (*serial clock*). O dispositivo com o qual o "*master*" comunica (o "*slave*") recebe dados no sinal MOSI (*master output slave input*) e responde dados no sinal MISO (*master input slave output*). Se CPOL=0 o estado de repouso do sinal SCK é zero, os *bits* transmitidos em ambos os sentidos são colocados nas respetivas linhas na transição ascendente de SCK e o dispositivo destino deve ler esse *bit* na transição descendente. Se CPOL=1 o significado do sinal SCK é o oposto do descrito.



[ref. MAXIM Appnote 4024]

[1,5 valores]

a) Pretende-se projetar um sistema digital que implemente o canal de receção SPI de um dispositivo "*slave*". Os 3 sinais de entrada envolvidos (SCK, MOSI e /SS) provêm de um sistema digital externo que opera com um sinal de relógio desconhecido, produzindo um sinal SCK de cerca de 500 kHz. O sinal de relógio local (do dispositivo "*slave*") será no mínimo 10 MHz e pretende-se desenvolver esse sistema como um circuito sequencial síncrono com esse sinal de relógio. Explique de que forma devem ser tratados aqueles sinais de entrada e quais as consequências que podem decorrer se isso não for feito de forma conveniente.

Como os 3 sinais externos referidos não têm qualquer referência temporal com o sinal principal de relógio, podem apresentar transições em qualquer instante de tempo em relação ao relógio principal. Se esses sinais forem usados como entradas de um circuito que determine o próximo estado de um conjunto de flip-flops, podem fazer com que os flip-flops capturem valores não coerentes devido à ocorrência de violações temporais que podem resultar em situações de meta-estabilidade ("captura" de um estado instável que aleatoriamente passa para 0 ou 1). A solução consiste em sincronizar esses sinais com o relógio global, fazendo-os passar por um registo de deslocamento de 2 andares. Desta forma obtém-se uma réplica dos sinais de entrada sincronizados com o relógio principal, embora com uma incerteza temporal que no máximo será igual a um período do relógio global. Como neste caso é dito que SCK tem uma frequência próxima de 500 kHz e o sinal global de relógio de 10 MHz (20 vezes mais rápido), essa incerta representará apenas 1/20 do período de SCK, o que é aceitável.

[3 valores]

b) Projete um módulo Verilog que implemente o canal de receção de um interface SPI "slave" para palavras de 8 *bits*, dentro das assunções referidas na alínea anterior. O bloco deve receber os 3 sinais originados por um dispositivo SPI "*master*" (SCK, MOSI, /SS) e produzir uma saída de 8 *bits* onde é apresentado o *byte* recebido (DOUT). Para sinalizar a receção de um novo *byte*, a saída DOUTEN deve ser ativada durante um ciclo do relógio principal (não SCK) sempre que é colocado em DOUT um novo byte recebido. O sinal global de *reset* deve ser síncrono e ativo ao nível lógico alto e todo o circuito deve ser síncrono com um sinal global de relógio de frequência não inferior a 10 MHz.

```
reg [7:0] SR; // input shift-register
reg mosir0, mosis, sckr0, sckr1, sckr2, ssbr0, ssbr1, ssbr2; //synch regs
wire ssbpe,     // positive pulse on the rising edge of /SS
     sck_pulse; // positive pulse on the rising or falling edge of SCK

// Two stage input synchronizers for SCK, SS and MOSI
always @(posedge clock)
begin
```

```
  mosir0 <= MOSI;
  mosis  <= mosir0; // MOSI synchronized to clock
  sckr0  <= SCK;
  sckr1  <= sckr0;  // SCK synchronized to clock
  sckr2  <= sckr1;  // one more flip-flop to generate the clock edge pulses
  ssbr0  <= SS;
  ssbr1  <= ssbr0;  // second stage of the synchronizer
  ssbr2  <= ssbr1;  // one more FF to delay ssbr1 one more clock period
end

assign ssbpe = ssbr1 & ~ssbr2; // positive pulse on the 0-1 edge of /SS

// Positive pulse on the 0-1 edge of SCK (CPOL==1) or 1-0 edge (CPOL==0)
assign sck_pulse = CPOL ? (sckr1 & ~sckr2) : (~sckr1 & sckr2);

// input shift-register
always @(posedge clock)
  if ( sck_pulse )
    SR <= {SR[6:0], mosis}; // shift left, add new bit in the right end

// output register:
always @(posedge clock)
begin
  DOUTEN <= ssbpe;  // DOUTEN is set to 1 when DOUT is loaded with fresh data
  if ( ssbpe )
    DOUT <= SR;     // Load current state of the input shift-register
end
```

[1,5 valores]

c) Admita que se pretende minimizar o consumo energético deste sistema, eliminando o requisito de utilizar um único domínio de relógio mas mantendo os sinais de saída DOUT e DOUTEN síncronos com o sinal global de relógio referido antes. Explique como orientaria neste caso o projeto desta unidade.

Se o circuito não tivesse de ser integralmente síncrono com o relógio global, a solução seria implementar o registo de deslocamento que agrupa os bits recebidos usando como relógio o sinal SCK. Como os sinais de saída DOUT e DOUTEN devem ser síncronos com o relógio global, bastaria sincronizar o sinal /SS e usar a sua transição positiva para capturar no registo DOUT o resultado na saída do registo de deslocamento.

Nesta questão não era exigida a presentação de uma solução completa que implemente esta versão. No entanto a dificuldade acrescida pelo facto do sinal SCK apenas existir durante a transmissão de dados justifica a presentação de uma possível implementação:

```
// SR is the input shift-register clocked with SCK (assuming CPOL=1)
reg [7:0] SR;
reg ssbr0, ssbr1, ssbr2;
wire ssbpe;
always @(posedge SCK)
  SR <= {SR[6:0], MOSI}; // shift left, add new bit in the right end

always @(posedge clock)
begin
  ssbr0 <= SS;    // two stage synchronizer (SS is the asynchronous /SS input)
  ssbr1 <= ssbr0;  // second stage of the synchronizer
  ssbr2 <= ssbr1;  // one more FF to delay ssbr1 one more clock period
  DOUTEN <= ssbpe; // DOUTEN is set to 1 when DOUT is loaded with fresh data
  if ( ssbpe )
    DOUT <= SR;    // Load current state of the input shift-register
end

assign ssbpe = ssbr1 & ~ssbr2; // positive pulse on the 0-1 edge of /SS
```

[5 valores]

4 - No projeto de um componente de um sistema de processamento de sinal é necessário implementar um bloco para calcular o módulo de um sinal representado pelas componentes real a_k e imaginária b_k, calculando a expressão:

$$y_k = \sqrt{a_k^2 + b_k^2}$$

Os sinais a, b e y são representados por inteiros de 10 bits e complemento para 2 e a frequência de amostragem é de 10 MHz. A saída pode ser produzida até 4 períodos de amostragem após a chegada das amostras de entrada. O sistema a implementar terá de operar com um sinal de relógio de frequência inferior a 200 MHz e pretende-se minimizar a área (ou complexidade lógica) do circuito. Numa reunião da equipa de projeto digital, 3 colegas apresentaram diferentes propostas (os nomes são fictícios):

T. *"Isto é muito simples, arranjei no opencores[1] um módulo em Verilog que calcula a raiz quadrada em 15 ciclos de relógio e que só tem 12 linhas de código; para calcular o argumento da raiz quadrada,* `yr`*, basta escrever "*`assign yr = a*a + b*b`*. Até já escrevi o módulo e verifiquei que funciona bem em simulação"*.

M. *"Também vi esse módulo para calcular a raiz quadrada, mas proponho que se utilize antes um único multiplicador sequencial porque usando um relógio de 200 MHz temos 20 ciclos de relógio por período de amostragem para realizar dois produtos de 10* bits *e o cálculo da raiz quadrada pode ser feito em* pipeline *no ciclo seguinte de amostragem"*.

E. *"Eu proponho uma solução mais simples, não precisa de multiplicadores e posso dizer que garante o resultado no tempo disponível..."*

[1,5 valores]

a) Represente o circuito que é gerado pela declaração "assign" proposta por T.

A declaração `assign` *referida representa (e sintetiza) um circuito totalmente combinatório que implementa a operação aritmética descrita, constituído por dois multiplicadores e um somador.*

[1,5 valores]

a) Porque é que a solução de T. foi imediatamente desprezada ?

Face aos requisitos enunciados, existem pelo menos 20 ciclos de relógio disponíveis (considerando o relógio de 10 MHz), pelo que não faz sentido calcular o argumento da raiz quadrada com um circuito combinatório. Se o cálculo da raiz quadrada requer 15 ciclos de relógio, então há 5 ciclos de relógio que podem ser aproveitados para reutilizar um único multiplicador e eventualmente recorrer até a um multiplicador sequencial em vez de combinatório. Para além disso, o comentário de T. "...e que só tem 12 linhas de código" também sugere que ele considera essa uma boa solução porque o código Verilog do calculador de raiz quadrada é pequeno, e como se sabe isso tem pouco a ver com a complexidade do circuito lógico que implementa.

[2 valores]

b) Coloque-se no lugar de E. e diga qual seria a solução que sugeriria, justificando as razões que a tornam melhor do que qualquer uma das outras.

*Para este problema a solução ideal seria recorrer ao algoritmo CORDIC, que numa das configurações (*vectoring mode*) permite determinar diretamente o módulo de um vetor sem requerer o cálculo explícito da função raiz quadrada. O algoritmo não necessita de multiplicadores e sua implementação sequencial realiza o cálculo num número de ciclos de relógio pouco maior do que o número de bits pretendido para o resultado (+ 3 ciclos). Apesar de ser necessário realizar um produto final por uma constante para compensar o ganho natural do algoritmo, isso poderia ser implementado ainda dentro dos 20 ciclos de relógio correspondentes a um período de amostragem.*

- FIM -

[1] `www.opencores.org`