UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Cursos: Bacharelado em Ciência da Computação e Bacharelado em Sistemas de Informação
Disciplina: Introdução aos Sistemas Lógicos
1º Semestre de 2019
Professor Luiz Filipe Menezes Vieira (*lfvieira@dcc.ufmg.br*)

Data de entrega: **24/06/2019**

Laboratório 4: Semáforo em Verilog

1 Introdução

Otávio não tem muita paciência com o semáforo da Avenida Carlos Luz. Um dia, distraído, ele esqueceu que anda malhando muito e não dosou sua força para apertar o botão do semáforo! Como resultado, a placa de controle que ficava na mesma caixa que o botão foi destruída. Como Otávio não quer encrenca e tem uma FPGA sobrando, ele pede que vocês o ajudem a fazer um novo circuito de controle antes que mais alguém descubra.

2 Descrição do Sistema

O sistema é formado pelo *push button bt* e pelos Semáforos A e B. bt, quando pressionado por um pedestre, fecha B para os carros sob as condições descritas na seção 2.2. O semáforo A controla o fluxo em um dos sentidos da via e o semáforo B controla o fluxo contrário, como ilustrado na figura 1.

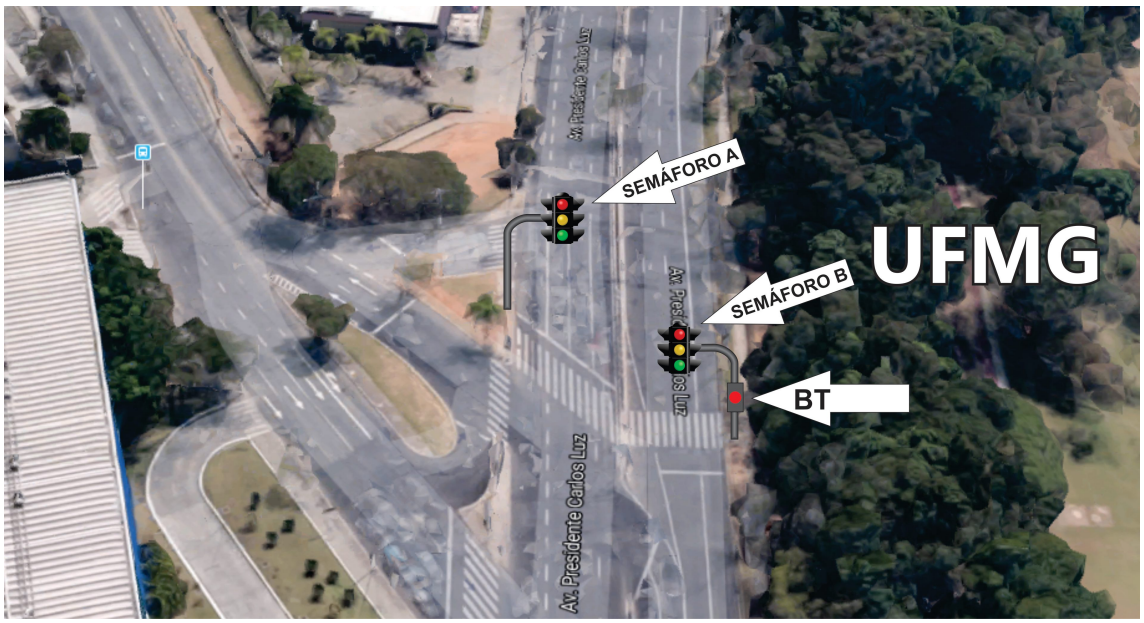

Figura 1: Esquema dos semáforos em questão

O estado de cada semáforo é representado com três bits utilizando a codificação *one-hot*, ou seja, a codificação em que os valores válidos correspondem àqueles em que apenas um dos bits permanece verdadeiro. Sendo assim, cada um dos três bits de cada semáforo é responsável por ativar uma das lâmpadas, na ordem ilustrada pela figura 2.

Você pode admitir que o sinal de *reset* será empregado no início de toda simulação, de forma que a inicialização dos registradores do sistema pode ser feita apenas quando houver uma **borda de descida do reset**. As subseções que se seguem detalham o comportamento de cada semáforo.

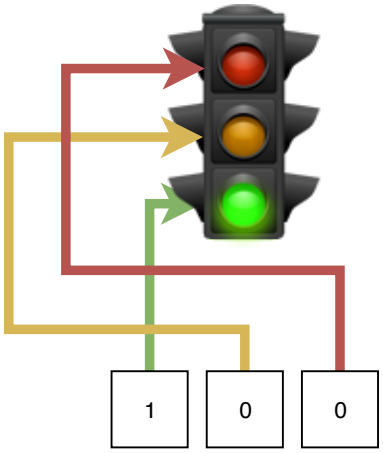

Figura 2: Correspondência entre os bits e as lâmpadas

2.1 Semáforo A

O comportamento do Semáforo A independe do estado de B, dependendo portanto apenas do estado do próprio A e da contagem do *clock*, além, é claro, do sinal de *reset*. Quando o *reset* é acionado, A passa para o estado verde (100) e, a partir daí, para o amarelo (010), vermelho (001) e novamente o verde, quando o ciclo é reiniciado. Cada estado é mantido por um período marcado por um determinado número de **bordas de subida do clock**. O número de bordas de subida que deve ser identificado para sair do estado verde e ir para o amarelo se encontra na constante `VERDE, por exemplo (veja o sumário dos nomes utilizados na interface na seção 3).

2.2 Semáforo B

O semáforo B, assim como o A, transiciona para o estado verde assim que o *reset* é ativado, permanecendo nesse estado enquanto o **sinal oriundo do botão não passa por uma borda de subida**. Assim que o botão é acionado, o semáforo B ainda aguarda o próximo ciclo de A para fechar. Sendo assim, no momento em que A passar de verde para amarelo, B também o fará. Da mesma forma, quando A passar de amarelo para vermelho, B o acompanhará sincronizadamente, e assim até ambos voltarem ao estado verde. Após isso, A seguirá sua sequência normal e B ficará preso no estado verde, aguardando uma nova borda de subida do sinal do botão. A borda de subida de *bt* é levada em consideração apenas quando B está verde.

3 Interface e Implementação

Os códigos serão avaliados utilizando a versão mais recente do *Icarus Verilog* em ambiente Linux. É recomendado que vocês utilizem condições similares para desenvolver seu projeto. Para instalar a versão mais recente do *Icarus* e do *GTKWave* (para visualização de forma de onda), utilize o seguinte comando:

sudo apt-get install verilog gtkwave

Junto dessa especificação, serão disponibilizados cinco arquivos. "semaforo.v" possui o módulo *semaforo* já declarado, indicando o local onde vocês devem inserir seu código. Não utilize outra interface ou um nome diferente do especificado, pois seu código será validado automaticamente. *clk* corresponde à entrada de *clock*, *rst* à entrada de *reset*, *bt* à entrada ligada ao botão, A à saída com os valores de estado do semáforo A e B à saída com o estado de B. Utilize as constantes `VERDE, `AMARELO e `VERMELHO para acessar as quantidades de bordas de subida do *clock* para as quais o semáforo A deve manter-se em cada estado. **Esses valores podem variar de 0 até 255.**

Cada arquivo "testbench_exemplo*.v" possui um *testbench*, ou seja, um arquivo utilizado para testar e depurar código *Verilog* em ambiente simulado (vejam a seção 4). Esses casos de teste são apenas ilustrativos e não possuem uma cobertura muito ampla, cabendo a vocês a responsabilidade de criar outros *testbenchs*. Para fazer isso, copiem e colem testbench_exemplo.v com o nome

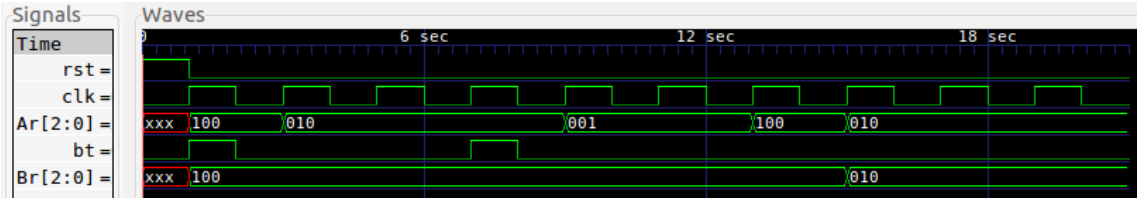

Figura 3: Resultado esperado da execução de "testbench_exemplo.v"

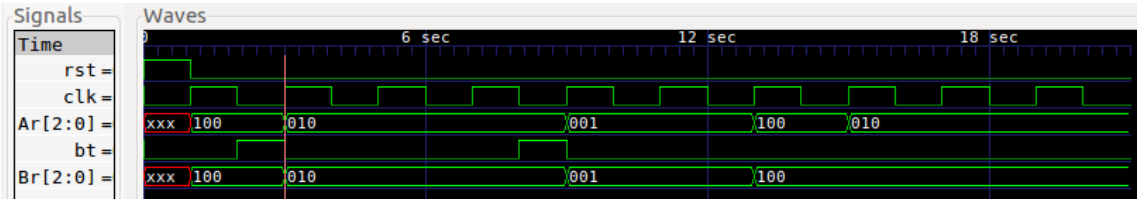

Figura 4: Resultado esperado da execução de "testbench_exemplo1.v"

desejado e alterem o nome do módulo de teste e o argumento de *dumpfile* conforme indicado pelos comentários. Em seguida, altere as sequências de valores de *clk*, *rst* e *bt* dentro dos blocos *initial* correspondentes. Vocês podem – e devem – testar outros valores das constantes declaradas no início do *testbench*.

Finalmente, há um *makefile* já pronto. Para testar seu código com um *testbench* de nome "nomeDoTestbench.v", abram o terminal do Linux no diretório onde estão os três arquivos e utilizem o comando:

make FILE=nomeDoTestbench

Para limpar os arquivos gerados, utilize:

make clean

Todos os inputs e outputs da interface representam os valores em *little endian*, ou seja, com o algarismo menos significativo à direita. Em *Verilog*, isso corresponde a declarar o registrador ou o fio nas formas *wire [n:0] nome*, para conexões de $n+1$ bits, e *reg [n:0] nome*, para registradores de $n+1$ bits. Para evitar erros, utilize essa codificação ao longo de todo o seu código.

4 Exemplos

Todos os *testbenches* fornecidos possuem a mesma forma de onda para o sinal de *reset*, além dos mesmos tempos de duração de cada estado de A. Sendo assim, as formas de onda relativas a A são iguais para todos os casos. O sinal *bt*, no entanto, é alterado de caso a caso, gerando formas de onda de B diferentes. Veja alguns comentários

"testbench_exemplo.v": (figura 3) Como a borda de subida do botão coincide com a borda de descida do *reset*, ela é desconsiderada. Sendo assim o sinal do semáforo B permanece verde até a segunda vez em que A se torna amarelo, quando acompanha o estado de A por um ciclo devido à segunda borda de subida do sinal do botão.

"testbench_exemplo1.v": (figura 4) Nesse caso a primeira borda de subida do sinal do botão é levada em consideração. O estado de B começa acompanhando o de A no primeiro ciclo, mas em seguida para no estado verde, já que a segunda borda de subida de *bt* ocorreu enquanto B estava amarelo.

"testbench_exemplo2.v": (figura 5) Nesse caso, o sinal de B acompanhou o de A durante toda a simulação, visto que o botão foi acionado sempre que B atingiu o estado verde e antes de A passar para o amarelo.

5 O que entregar?

Cada grupo deve submeter **APENAS UM ZIP** contendo, na raiz,

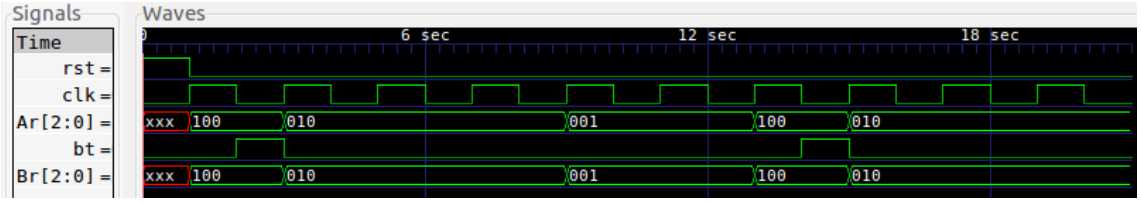

Figura 5: Resultado esperado da execução de "testbench_exemplo2.v"

1. Documentação (em pdf) breve contendo decisões de implementação, as máquinas de estado utilizadas como referência para a implementação do código e a descrição dos *testbenchs* implementados,
2. O arquivo "semaforo.v" e
3. Os arquivos de *testbench* implementados.

6 Considerações importantes

- A legibilidade do código será considerada na correção! Use nomes intuitivos, indente seu código e faça bom uso de comentários.
- Fique atento quanto ao uso de eventos do bloco *always*. Trocar uma borda de descida por uma de subida gerará defasagem na forma de onda e deixará sua resposta incorreta.
- Tome cuidado com as atribuições assíncronas (<=). O valor considerado para a atribuição será o da ultima execução de *always*!
- Qualquer *string* binária pode ser considerada um sinal de *clock*. Sendo assim, teste sinais de *clock* diferentes do clássico 010101010101010, ou seja, sinais com frequências inconstantes, como 00010111, por exemplo.
- Como mencionado na seção 3, a interface utiliza codificação *little endian*. É expressamente recomendado que você utilize essa codificação em todas as conexões e registradores do sistema.
- Para fazer as máquinas de estado, você pode utilizar qualquer editor. Sugerimos pesquisar sobre o sistema *online draw.io*.
- **Apenas um membro do grupo deve submeter o trabalho com o nome de todos do grupo.**