



Departamento de Engenharia Elétrica
Universidade de Brasília
Disciplina: Laboratório de Sistemas Digitais (Turma: 04)
Professor: Guilherme Torres

Primeiro Semestre de 2024

Experimento 8

Datas da realização do experimento: 11 de junho de 2024

Aluno: Felipe Lopes Gibin Duarte

Matrícula: 231025207

1. INTRODUÇÃO

Neste experimento, vamos expandir sobre os conceitos aprendidos nos experimentos anteriores ao implementar: um contador módulo 100 usando 2 contadores módulo 10, um sistema de temporização e um sistema de controle de semáforos, tudo isso usando a linguagem de descrição de hardware VHDL. Neste experimento, usaremos boa parte dos conceitos aprendidos durante o semestre, sejam eles referentes à disciplina da teoria ou ao laboratório. Desse modo, todos os procedimentos serão desenvolvidos usando o software “ModelSim”. Para alcançar tais objetivos, o seguinte foi feito.

2. DESENVOLVIMENTO

QUESTÃO 01.

Implementar um contador módulo 100 com saída BCD, com entradas de 'reset' e 'load' síncronas, clock de 1 Hz e entrada de 'enable' (ativa em nível baixo). A caixa preta e o diagrama de blocos da arquitetura interna desse sistema são apresentados na Figura 1.

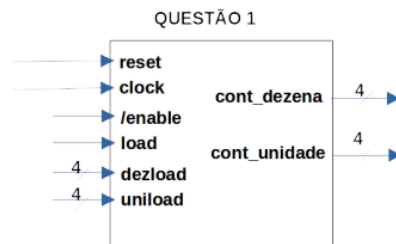


Figure 1: Caixa preta (a) do sistema a ser implementado na questão 1.

A entidade contador100 (Figura 2a) implementa o contador de 100 tempos em si. Trata-se de um contador que, a cada ciclo de clock, conta de 0 a 99 em representação BCD, isto é, com duas saídas de 4 bits que representam, respectivamente, o dígito da dezena e o da unidade. Para implementar este contador de módulo 100, você deve utilizar (como componentes) dois contadores de módulo 10. Pesquise sobre como cascatear contadores de módulo 10 (Figura 2b) de modo a construir contadores de módulo 100, módulo 1000, etc.

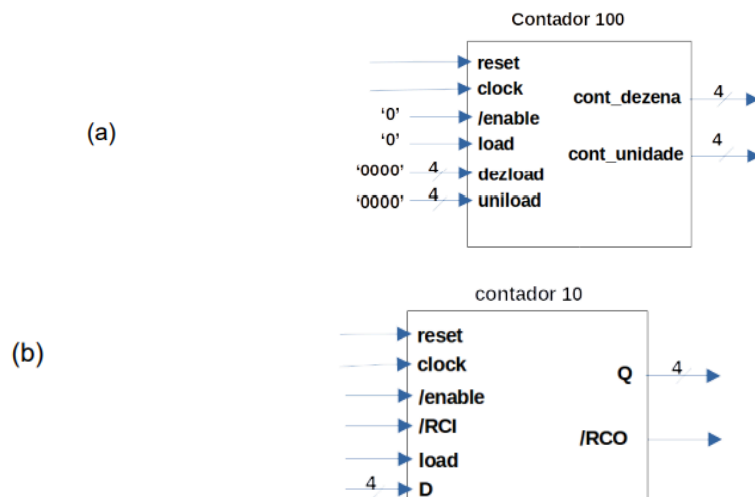


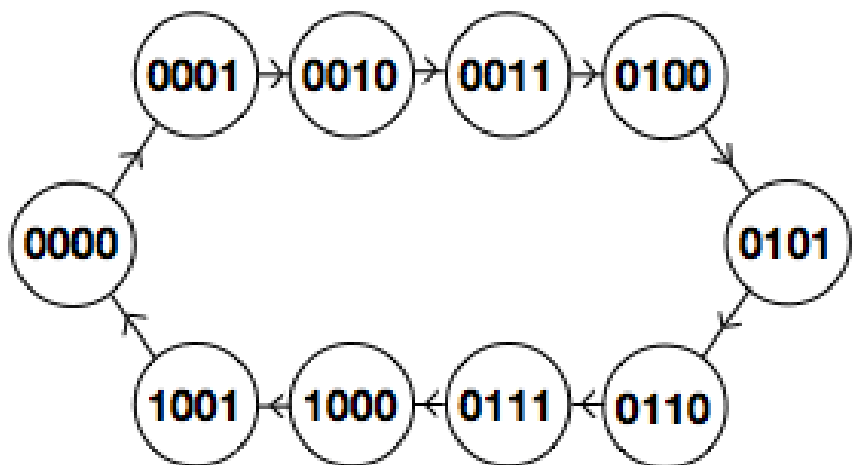
Figura 2: Caixas pretas das entidades contador100 (a) e contador10 (b).

O contador de módulo 10 será implementado pela entidade contador10, cuja caixa preta é mostrada na Figura 2b. Este contador de módulo 10, deve ser implementado como uma máquina de estados do tipo **Moore**. Essa máquina terá 10 estados, cada um associado a uma saída Q de 4 bits: de "0000" (decimal 0) até "1001" (decimal 9). As transições de estado serão controladas pelas variáveis de entrada reset (que, de forma síncrona, leva a máquina de volta ao estado inicial se reset = '1'), enable e RCI (que são ativas em nível baixo, ou seja, a contagem está ativada quando enable = '0' e RCI = '0') (RCI vem da sigla em inglês ripple carry-in), e load e D (de forma síncrona, a máquina deve ser levada ao estado indicado por D se load = '1'). A saída RCO (sigla do inglês ripple carry-out) é ativa em nível baixo, e deverá ser '0' se e somente se Q = '1001', caso contrário será '1'. Esta saída usada para cascatear

contadores de módulo 10, de modo a construir contadores de módulo 100, módulo 1000, etc. A ação de 'reset' deve ter prioridade sobre a ação de 'load', que por sua vez deve ter prioridade sobre a contagem.

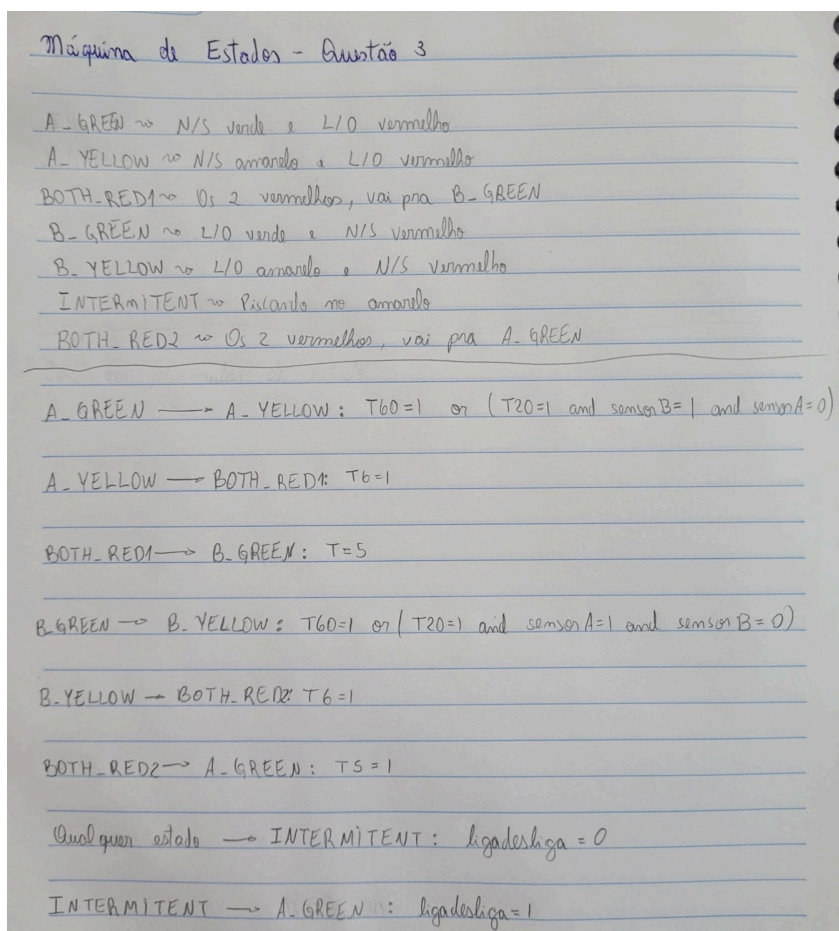
-Detalhes do projeto

Diagrama de estados do contador mod 10:



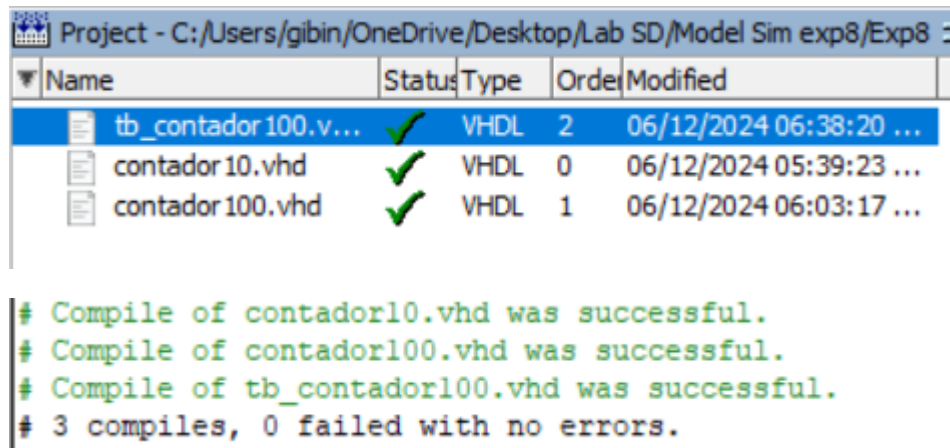
www.f-alpha.net

Funcionamento da máquina de estados:



-O código documentado está no segundo arquivo

-Confirmação do código compilado:

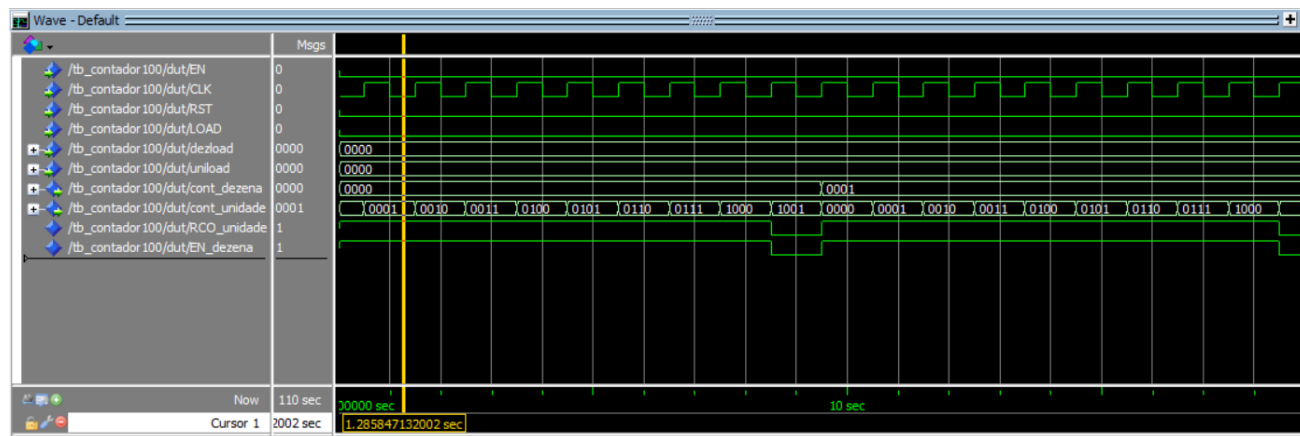


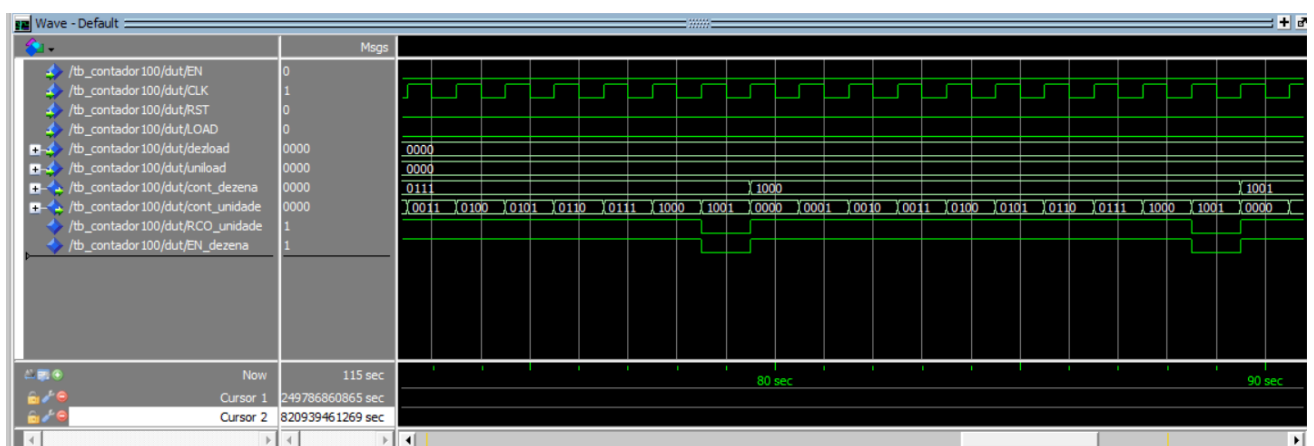
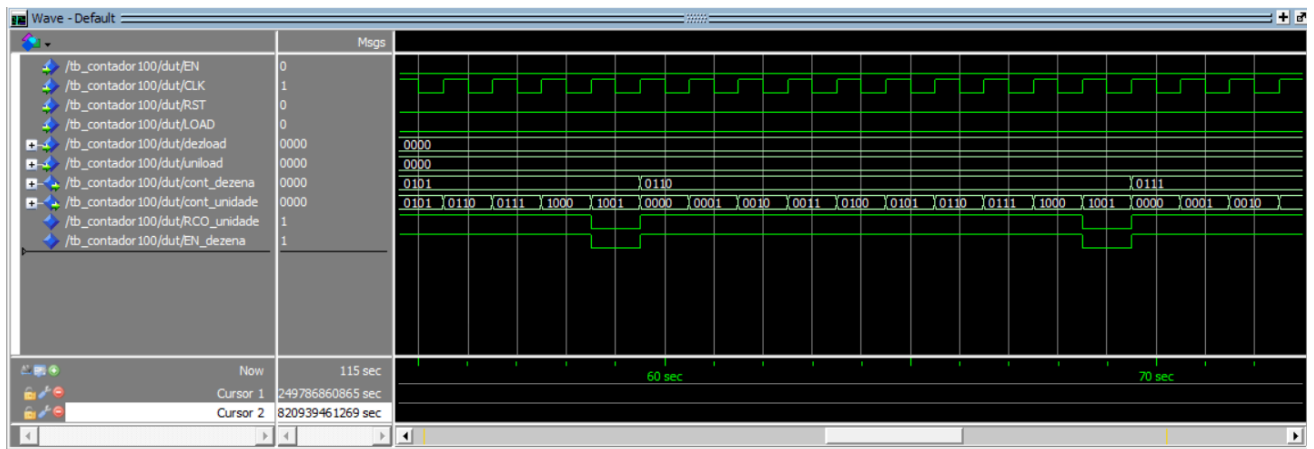
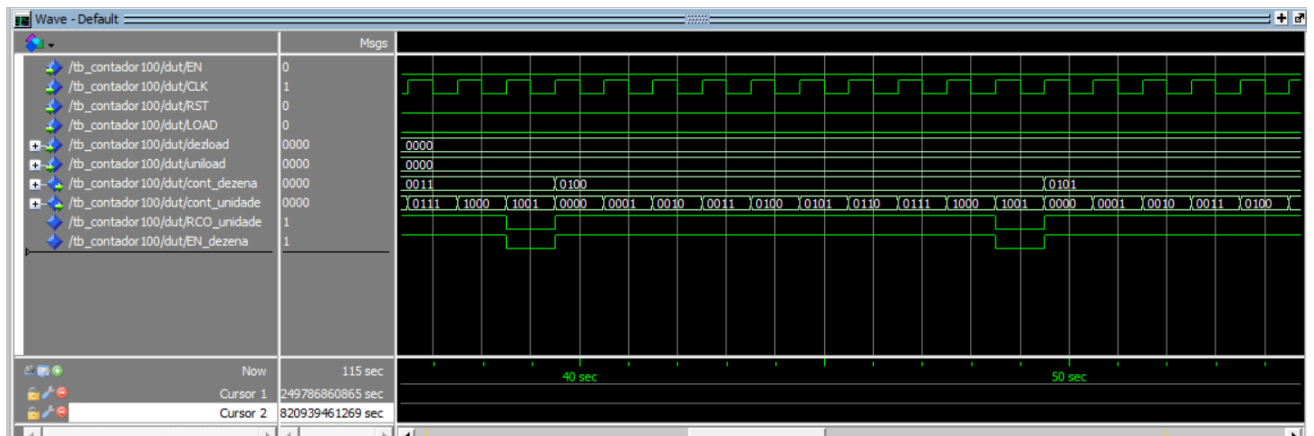
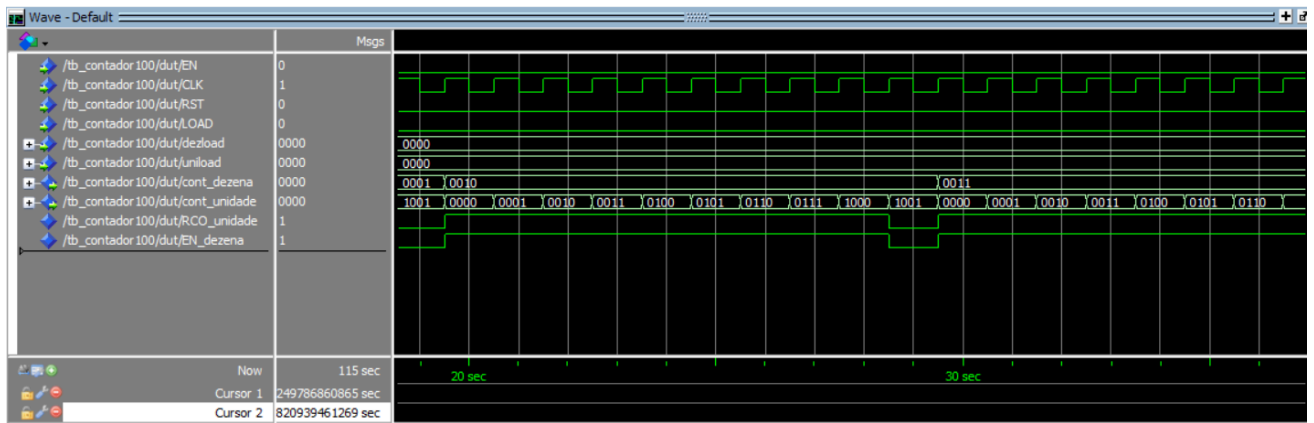
The screenshot shows the ModelSim project window for 'C:/Users/gibin/OneDrive/Desktop/Lab SD/Model Sim exp8/Exp8'. It lists three files: 'tb_contador100.v...', 'contador10.vhd', and 'contador100.vhd', all with a status of 'Success' (indicated by green checkmarks) and type 'VHDL'. Below the file list, the compilation log shows successful compilation for all three files and a summary: '# 3 compiles, 0 failed with no errors.'

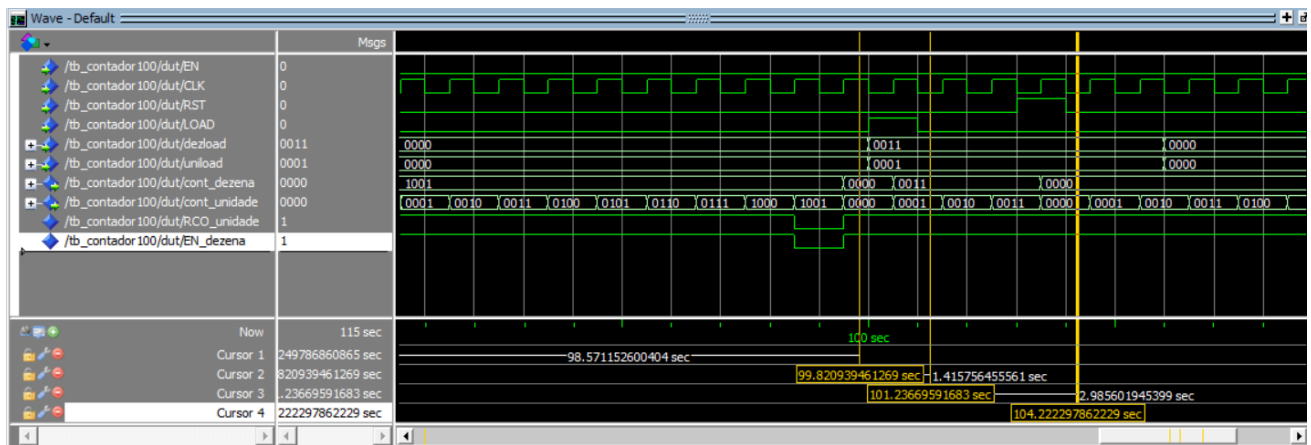
Name	Status	Type	Order	Modified
tb_contador100.v...	✓	VHDL	2	06/12/2024 06:38:20 ...
contador10.vhd	✓	VHDL	0	06/12/2024 05:39:23 ...
contador100.vhd	✓	VHDL	1	06/12/2024 06:03:17 ...

```
# Compile of contador10.vhd was successful.
# Compile of contador100.vhd was successful.
# Compile of tb_contador100.vhd was successful.
# 3 compiles, 0 failed with no errors.
```

-Simulação do código no ModelSim:







-Análise:

Cursor 1) Começa contagem

Cursor 2) Termina contagem (após 100 subidas do clock)

Cursor 3) Load, vai para 31 (0011, 0001)

Cursor 4) Reset

QUESTÃO 02.

Implemente um sistema de temporização do controle de semáforos, com 'reset' síncrono, clock de 1Hz, quatro saídas binárias, indicando que já se passaram, respectivamente, 5 segundos (T5), 6 segundos (T6), 20 segundos (T20) e 60 segundos (T60). A caixa preta da arquitetura interna desse sistema é apresentada na Figura 3(a).

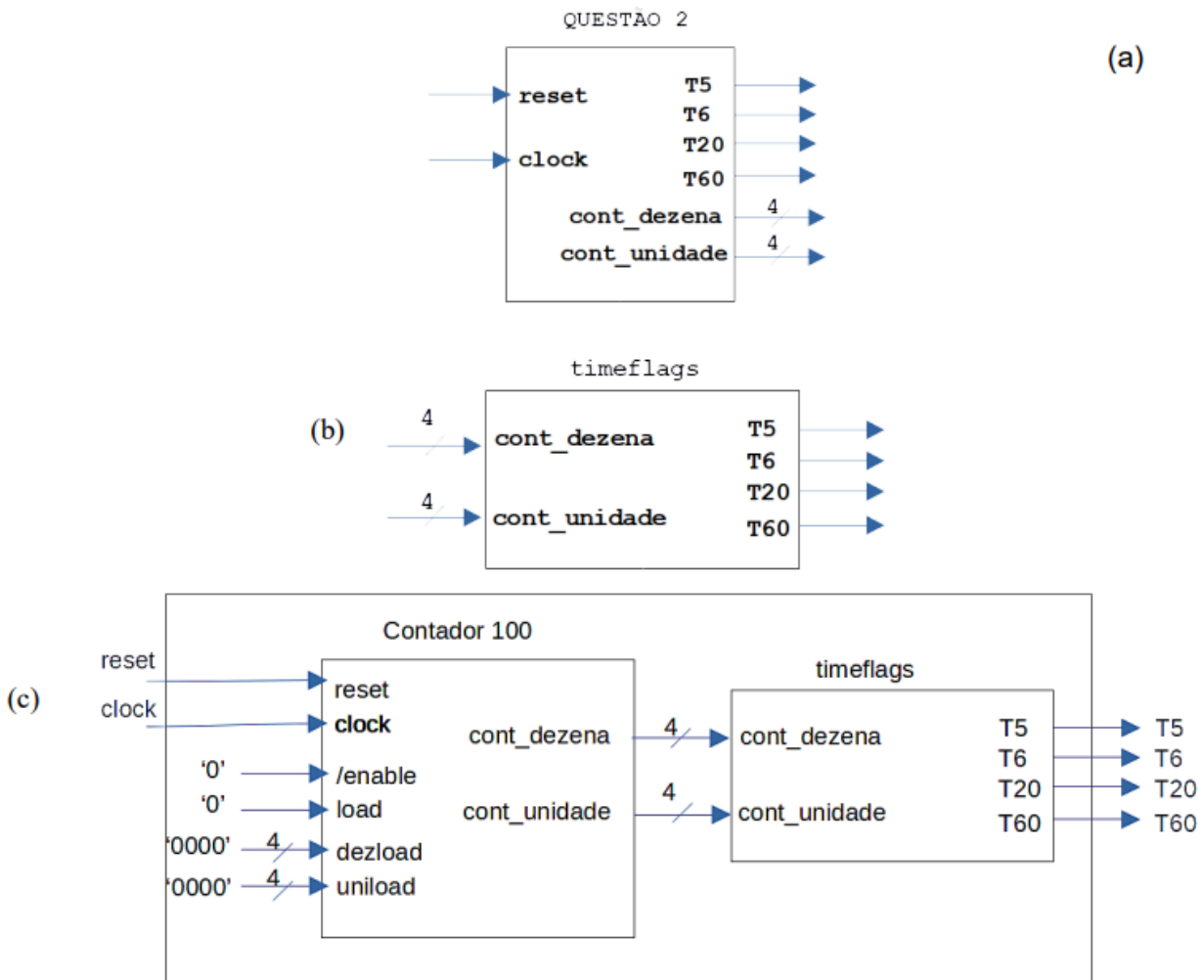


Figure 3: Caixa preta (a) e diagrama de blocos da arquitetura interna (b) e sistema a ser implementado na questão 2 (c).

Na Figura 3(c) é apresentado o sistema completo a ser implementado. A entidade timeflags (Figura 3b) é responsável por verificar se já se passaram, respectivamente, 5 segundos (T5), 6 segundos (T6), 20 segundos (T20) e 60 segundos (T60). As entradas são dois números em representação BCD, dezena e unidade, e as saídas são os quatro flags indicadores: T5, T6, T20 e T60.

Esses flags indicadores podem ser implementados usando atribuições condicionais (estrutura "when-else") e os operadores de comparação da linguagem VHDL, conforme exemplificado a seguir. Trata-se de um circuito puramente combinacional, não sendo necessário o uso de uma estrutura "process".


```

Ya <= '1' when (A > x"35") else -- operador "maior que"
    '0';

Yb <= '1' when (A >= x"35") else -- operador "maior ou igual"
    '0';

Yc <= '1' when (A < x"35") else -- operador "menor que"
    '0';

Yd <= '1' when (A <= x"35") else -- operador "menor ou igual"
    '0';

Ye <= '1' when (A = x"35") else -- operador "igual a"
    '0';

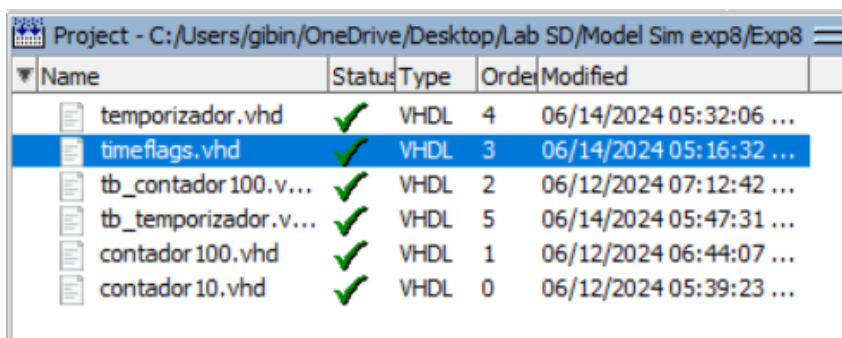
Yf <= '1' when (A /= x"35") else -- operador "diferente de"
    '0';

```

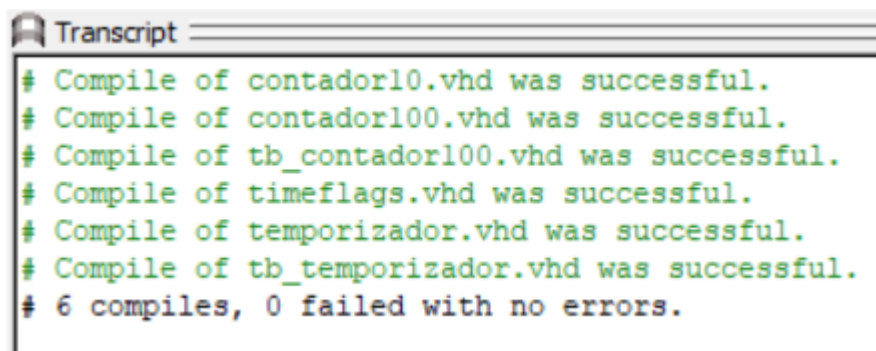
Nos exemplos acima, x"35" é a representação em hexadecimal do número binário de 8 bits "00110101", o qual, em representação BCD, corresponde ao decimal 35, pois os quatro primeiro bits ("0011") correspondem ao número decimal 3 e os quatro últimos ("0101") correspondem ao número decimal 5. Você pode usar a representação hexadecimal para evitar escrever longas sequências de bits. Por exemplo, o número x"13F7" corresponde ao número binário "0001001111110111".

-O código documentado está no segundo arquivo

-Confirmação do código compilado:



Name	Status	Type	Order	Modified
temporizador.vhd	✓	VHDL	4	06/14/2024 05:32:06 ...
timeflags.vhd	✓	VHDL	3	06/14/2024 05:16:32 ...
tb_contador100.v...	✓	VHDL	2	06/12/2024 07:12:42 ...
tb_temporizador.v...	✓	VHDL	5	06/14/2024 05:47:31 ...
contador100.vhd	✓	VHDL	1	06/12/2024 06:44:07 ...
contador10.vhd	✓	VHDL	0	06/12/2024 05:39:23 ...

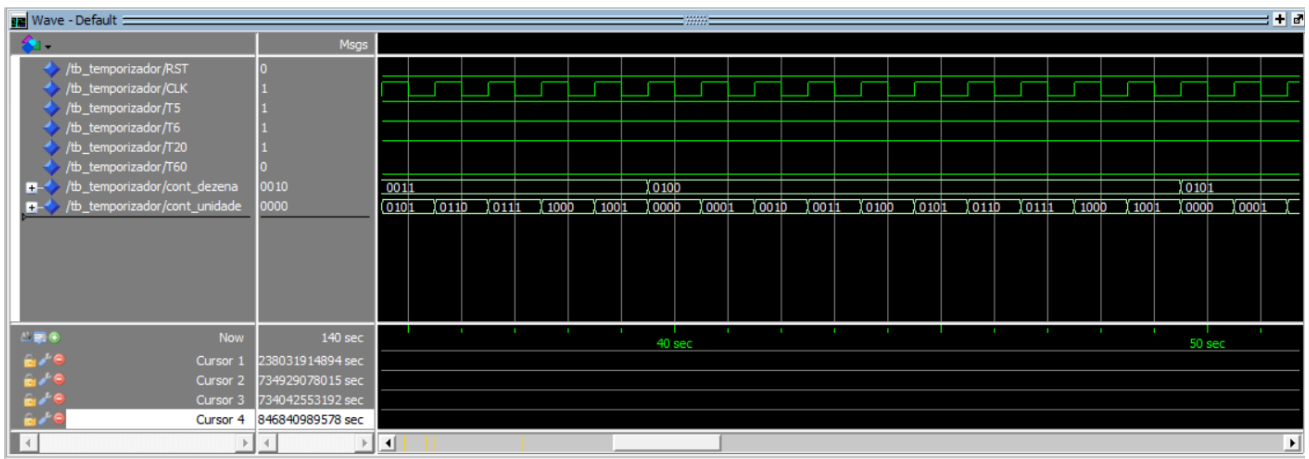
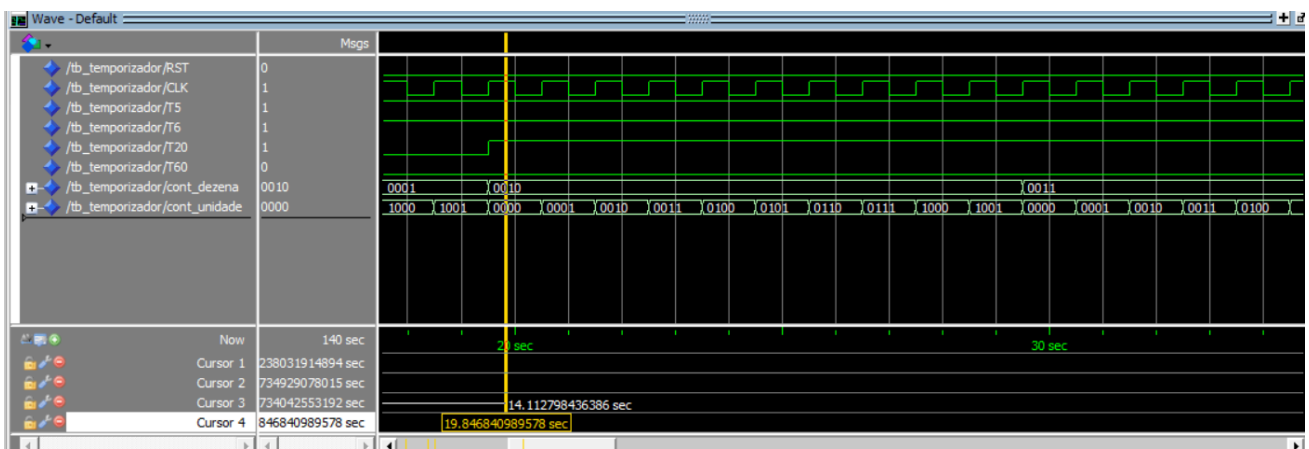
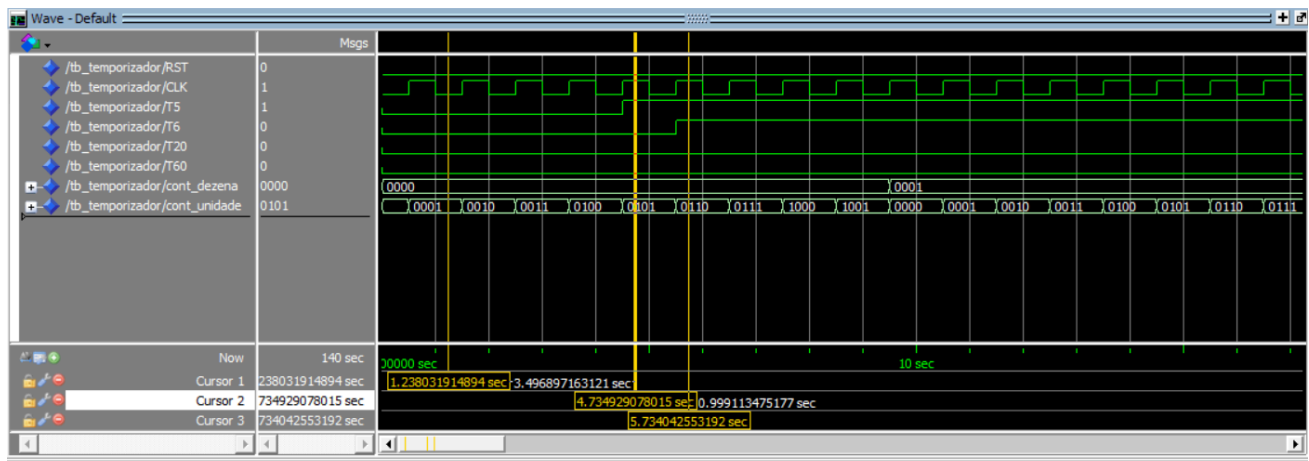


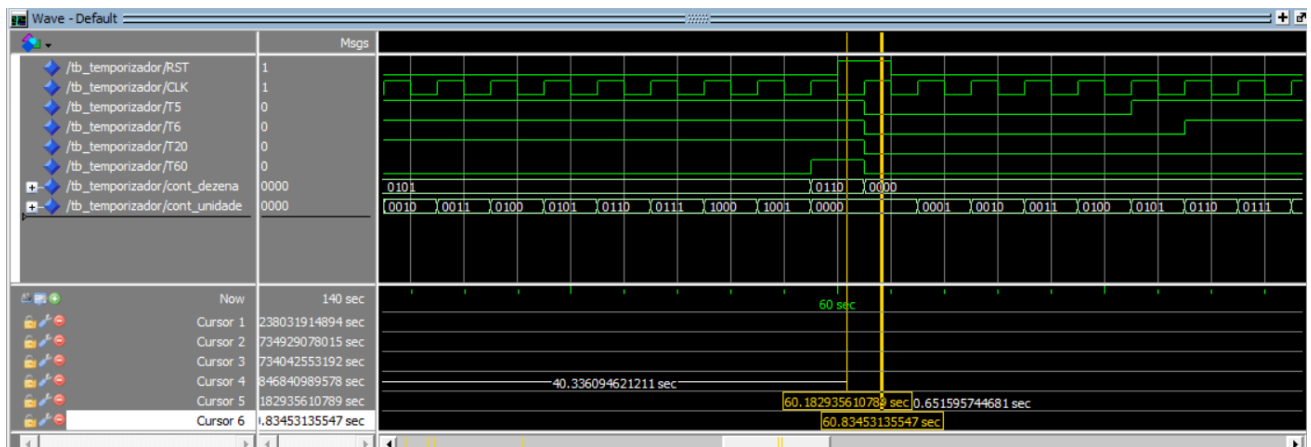
```

# Compile of contador10.vhd was successful.
# Compile of contador100.vhd was successful.
# Compile of tb_contador100.vhd was successful.
# Compile of timeflags.vhd was successful.
# Compile of temporizador.vhd was successful.
# Compile of tb_temporizador.vhd was successful.
# 6 compiles, 0 failed with no errors.

```


-Simulação do código no ModelSim:





-Análise:

Cursor 1) Começa contagem

Cursor 2) Flag T5='1', passaram-se 5 segundos

Cursor 3) Flag T6='1', passaram-se 6 segundos

Cursor 4) Flag T20='1', passaram-se 20 segundos

Cursor 5) Flag T60='1', passaram-se 60 segundos (termina contagem)

Cursor 6) Reset

QUESTÃO 03.

Implemente o sistema de controle de semáforos com três bits de entrada (sensor de carros na direção norte/sul, sensor de carros na direção leste/oeste e chave de liga/desliga do sistema) e saídas representando os estados dos semáforos e o temporizador/contador (luzes verde, amarela e vermelha de cada direção e estado do contador). O cruzamento é ilustrado na Figura 4.

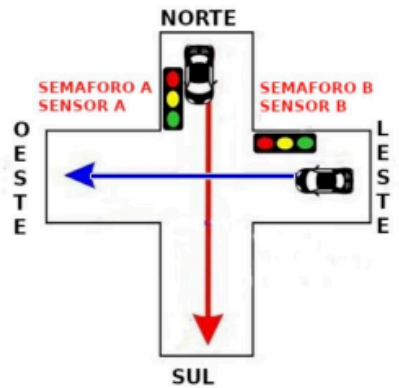


Figura 4: Ilustração do cruzamento, mostrando a posição dos semáforos e dos sensores.

O cruzamento deverá ficar liberado para determinada direção (luz verde) e bloqueado na outra direção (luz vermelha) por 60 segundos, exceto caso haja carro esperando na direção oposta e não haja carros atravessando na direção em questão e já se tenha passado pelo menos 20 segundos. Antes de trocar a pista liberada, o semáforo deverá mostrar a luz amarela (com luz vermelha no outro semáforo) por 6 segundos e, em seguida, a luz vermelha (em ambos os semáforos) por 5 segundos. A qualquer momento, se a chave de liga/desliga for desativada, os semáforos deverão entrar em estado intermitente, no qual alternarão entre a luz amarela e nenhuma luz acesa, com intervalo de 1 segundo entre cada, devendo voltar a qualquer momento ao funcionamento normal caso a chave seja reativada.

Mais especificamente, o sistema deve ser implementado com as seguintes entradas:

- sensorA: 1 bit que indica se há carros na direção norte/sul),
- sensorB: 1 bit que indica se há carros na direção leste/oeste),
- Chave ligadesliga: 1 bit que indica se chave de liga/desliga do sistema foi acionada;

e as seguintes saídas:

- semaforoA: 3 bits indicando quais das luzes (vermelha, verde e amarela) do semáforo da direção norte/sul estão acesas;
- semaforoB: 3 bits indicando quais das luzes (vermelha, verde e amarela) do semáforo da direção leste/oeste estão acesas;
- contador de tempo: 4 bits para representar as unidades e 4 bits para representar as dezenas do contador de tempo implementado nas questões anteriores.

A caixa preta desse sistema é apresentada na Figura 5a. Serão usadas (como componentes) todas as entidades utilizadas na arquitetura interna da entidade da questão 2 e uma nova entidade: maqestados, que será detalhada a seguir.

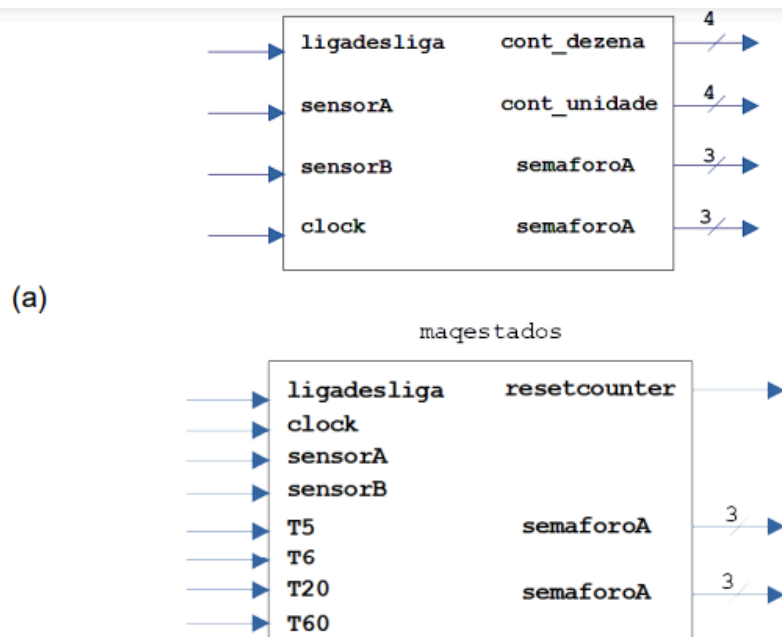


Figure 5: Caixas pretas das entidades exp8visto3 (a) e entidade maquestados (b).

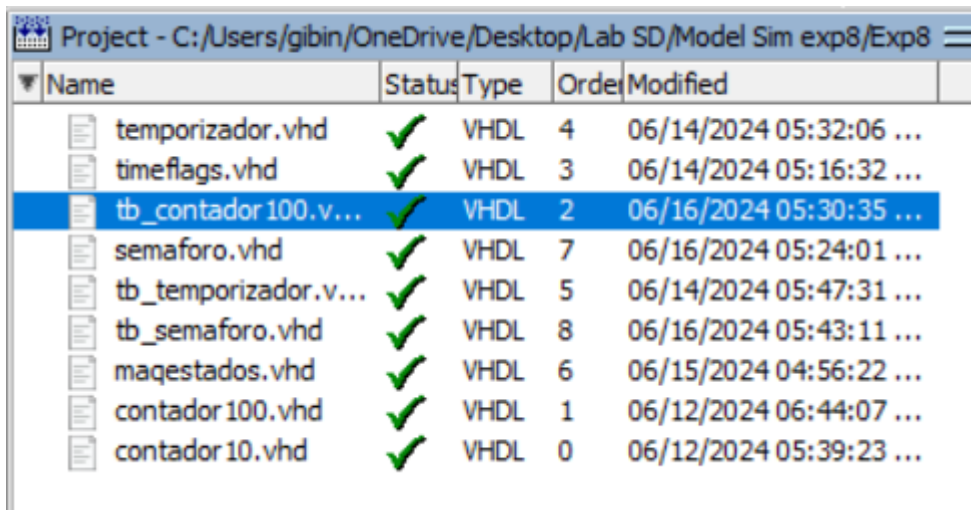
A entidade maquestados é a responsável por implementar o controle dos semáforos. A caixa preta dessa entidade é apresentada na Figura 5b. Deve ser implementada como uma máquina de estados, com duas saídas do tipo Moore — dois vetores de 3 bits semaforoA e semaforoB, que indicarão se as luzes vermelho, amarelo e verde dos semáforos das pistas norte/sul e leste/oeste, respectivamente, estarão acesas ou não (1 bit para cada cor, nessa ordem) — e uma saída do tipo Mealy — um sinal de 1 bit resetcounter, que indicará quando o contador de tempo deverá ser reinicializado.

As transições de estado serão controladas pelas variáveis de entrada ligadesliga, sensorA, sensorB, T5, T6, T20 e T60. As três primeiras serão associadas às entradas de mesmo nome da entidade “top level” (Figura 5(a)) e as quatro últimas às saídas da entidade timeflags. O cruzamento deverá ficar liberado para determinada direção (luz verde) e bloqueado na outra direção (luz vermelha) por 60 segundos, exceto caso haja carro esperando na direção oposta e não haja carros atravessando na direção em questão e já se tenha passado pelo menos 20 segundos. Antes de trocar a pista liberada, o semáforo deverá mostrar a luz amarela (com luz vermelha no outro semáforo) por 6 segundos e, em seguida, a luz vermelha (em ambos os semáforos) por 5 segundos. A qualquer momento, se a chave de liga/desliga for desativada, os semáforos deverão entrar em estado intermitente, no qual alternarão entre a luz amarela e nenhuma luz acesa, com intervalo de 1 segundo entre cada, devendo voltar a qualquer momento ao funcionamento normal caso a chave seja reativada.

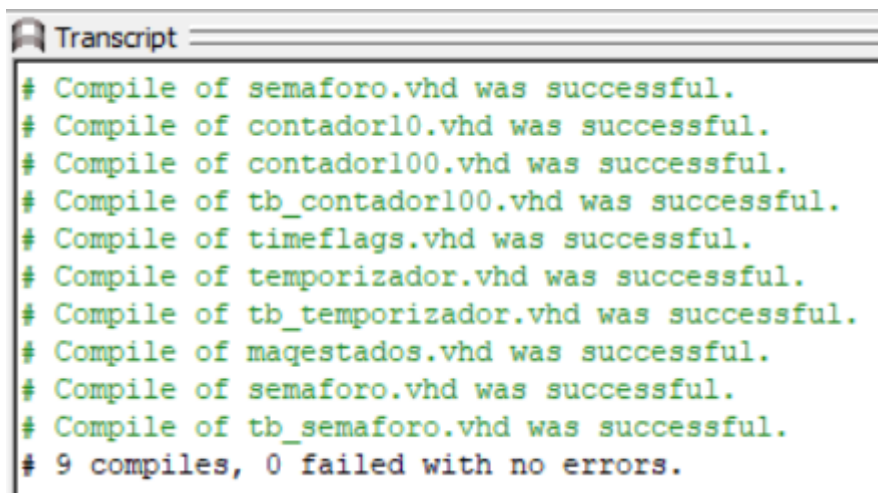
Note que a entidade maquestados deve, por meio da saída resetcounter, reinicializar o contador de tempo (implementado pela entidade contador100), isto é, fazer a saída resetcounter = ‘1’, sempre que a máquina estiver se preparando para mudar de estado, isto é, se o estado seguinte (nextState) for diferente do estado atual (currentState). Note que esta é uma saída do tipo Mealy, pois depende não só do estado atual, mas também das variáveis de entrada (que efetivamente determinam o estado seguinte). Isto pode ser implementado com uma lógica combinacional de estado seguinte, atribuindo um valor (‘0’ ou ‘1’) a resetcounter sempre que um valor for atribuído à variável de estado seguinte (nextState). A implementação de máquinas de estado Mealy em VHDL discutida em maiores detalhes no documento “Implementando máquinas de estados síncronas do tipo Mealy em VHDL”.

-O código documentado está no segundo arquivo

-Confirmação do código compilado:

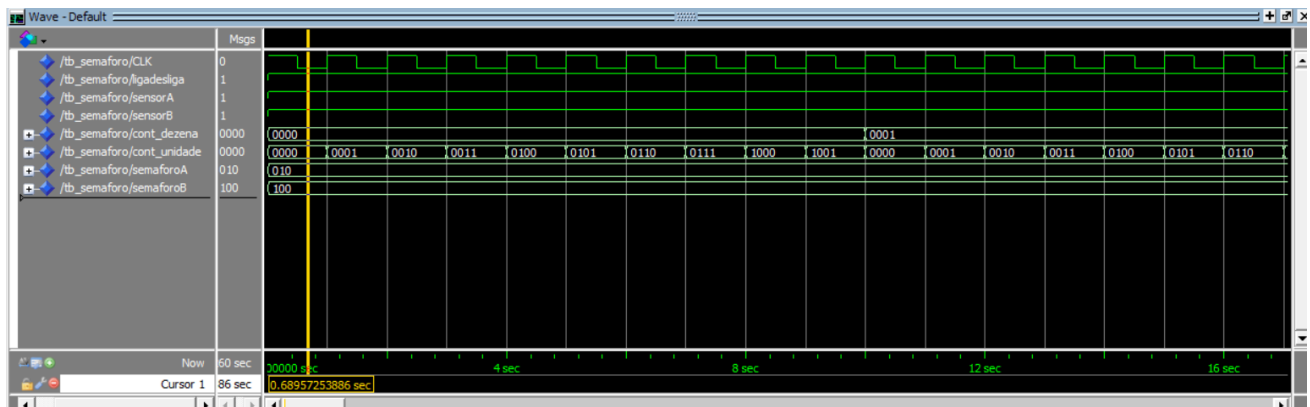


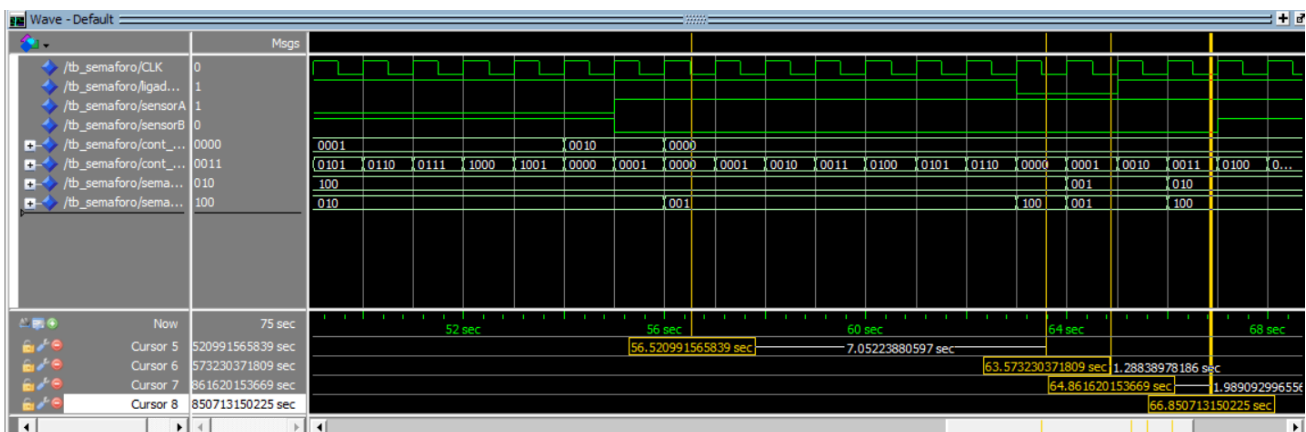
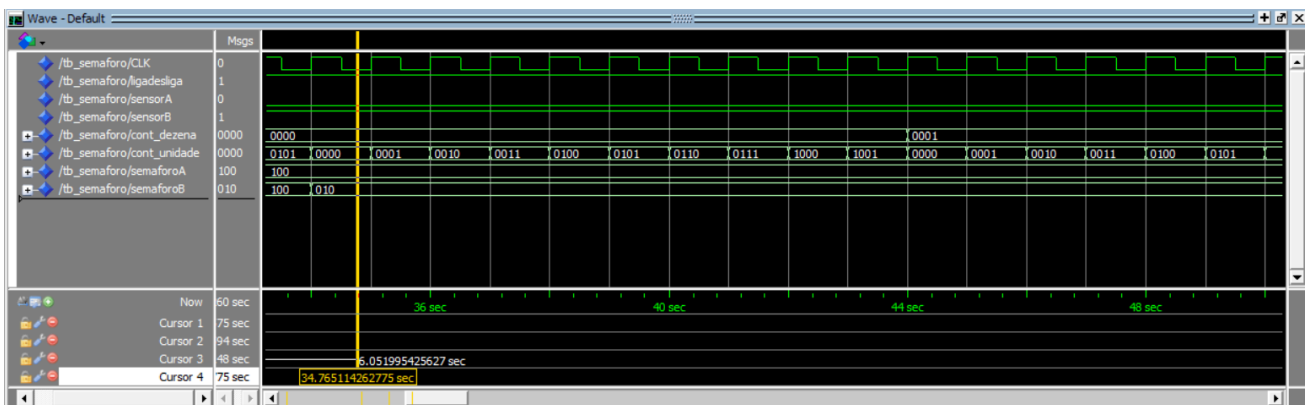
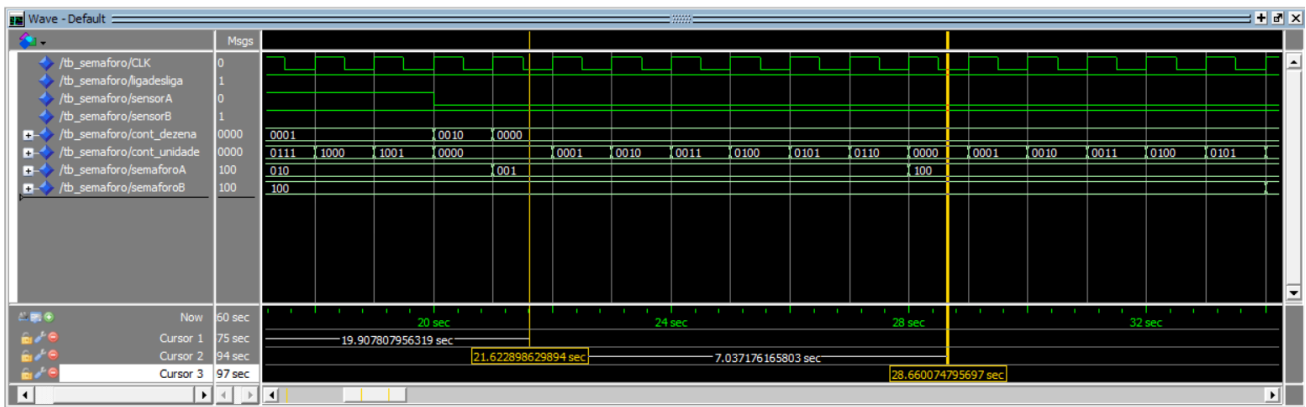
Name	Status	Type	Order	Modified
temporizador.vhd	✓	VHDL	4	06/14/2024 05:32:06 ...
timeflags.vhd	✓	VHDL	3	06/14/2024 05:16:32 ...
tb_contador100.v...	✓	VHDL	2	06/16/2024 05:30:35 ...
semaforo.vhd	✓	VHDL	7	06/16/2024 05:24:01 ...
tb_temporizador.v...	✓	VHDL	5	06/14/2024 05:47:31 ...
tb_semaforo.vhd	✓	VHDL	8	06/16/2024 05:43:11 ...
maquestados.vhd	✓	VHDL	6	06/15/2024 04:56:22 ...
contador100.vhd	✓	VHDL	1	06/12/2024 06:44:07 ...
contador10.vhd	✓	VHDL	0	06/12/2024 05:39:23 ...



```
# Compile of semaforo.vhd was successful.
# Compile of contador10.vhd was successful.
# Compile of contador100.vhd was successful.
# Compile of tb_contador100.vhd was successful.
# Compile of timeflags.vhd was successful.
# Compile of temporizador.vhd was successful.
# Compile of tb_temporizador.vhd was successful.
# Compile of maquestados.vhd was successful.
# Compile of semaforo.vhd was successful.
# Compile of tb_semaforo.vhd was successful.
# 9 compiles, 0 failed with no errors.
```

-Simulação do código no ModelSim:





-Análise:

Cursor 1) ligadesliga = 1, sensorA = 1, sensorB = 1, semaforoA = 010 (verde),
semaforoB = 100 (vermelho);

Cursor 2) ligadesliga = 1, sensorA = 0, sensorB = 1, semaforoA = 100 (vermelho),
semaforoB = 100 (vermelho);

Cursor 3) ligadesliga = 1, sensorA = 0, sensorB = 1, semaforoA = 001 (amarelo),
semaforoB = 100 (vermelho);

Cursor 4) ligadesliga = 1, sensorA = 0, sensorB = 1, semaforoA = 100 (vermelho),

semaforoB = 010 (verde);

Cursor 5) ligadesliga = 1, sensorA = 1, sensorB = 0, semaforoA = 100 (vermelho),
semaforoB = 001 (amarelo);

Cursor 6) ligadesliga = 1, sensorA = 1, sensorB = 0, semaforoA = 100 (vermelho),
semaforoB = 100 (vermelho);

Cursor 7) ligadesliga = 0, sensorA = 1, sensorB = 0, semaforoA = 001 (amarelo),
semaforoB = 001 (amarelo);

Cursor 8) ligadesliga = 1, sensorA = 1, sensorB = 0, semaforoA = 010 (verde),
semaforoB = 100 (vermelho);

3. CONCLUSÃO

Após todos os procedimentos, pudemos consolidar ainda mais os conceitos essenciais de design digital utilizando VHDL. Nesse experimento em particular, construímos um semáforo, em que antes foi preciso fazer um contador mod 100, um temporizador e uma máquina de estados. A complexidade deste experimento foi significativamente maior, em que, por ser o último experimento do semestre, foram utilizados todos os conceitos aprendidos durante o mesmo. Os maiores obstáculos encontrados no experimento foram projetar a máquina de estados e montar um testbench que testasse as funcionalidades do semáforo na questão 3. Por fim, a experiência foi proveitosa ao juntar, em um experimento só, a maioria dos conceitos de VHDL aprendidos durante o semestre letivo.