Insper

Elementos de Sistema - Projeto C - Lógica Combinacional

Rafael Corsi - rafael.corsi@insper.edu.br

Março - 2018

```
Prazo original : Quarta Feira - 14/3/2018
Atrasado até   : Quarta Feira - 21/3/2018
```

Descrição

Esse projeto tem como objetivo trabalhar com portas lógicas e sistemas digitais combinacionais (sem um clock) em FPGA e VHDL. Os elementos lógicos desenvolvidos nessa etapa serão utilizados como elementos básicos para a construção do computador.

O desenvolvimento será na linguagem VHDL com o git. Os alunos deverão se organizar para implementar todos os elementos propostos. O facilitador escolhido será responsável pela completude e consistência do branch master do fork do grupo.

Integrantes

Este projeto é para ser realizado por todos os integrantes do grupo. Tarefas devem ser criadas no Trello e alunos poderão executar elas em duplas, pratique pair-programming, porém evite sempre trabalhar com a mesma dupla, ou seja, cada projeto deverá ser realizado por uma nova dupla. Se organizem para tentar desenvolver o projeto de forma que todos aprendam o máximo possível, embora só um integrante de uma dupla possa fazer os commits para o github. Converse com os colegas a respeito do projeto, mas não aceite código pronto.

Controle de Tarefas e Repositório

Atualize seu Fork do repositório do projeto do github:

- https://github.com/Insper/Z01

Nas discussões com os outros colegas o scrum master deve definir os módulos que você irá desenvolver. Crie uma rotina para commits e pull-request. Sempre teste os módulos e verifique se está fazendo o esperado.

Facilitador (Scrum Master)

- Organizar o github + issues + project
- Fazer a atualização do fork com o upstream
- Gerenciar o grupo (atribuir tarefas)
- Aceitar os pull-requests
- Criar relatório da performance de cada um do grupo
- Entregar/Apresentar o projeto no final

Desenvolvedores

- Realizar as tarefas atribuidas pelo scrum-master
- Realizar os pull-requests
- Testar os códigos

Intruções

A pasta contém dois diretórios distintos : src/ e Quartus/. O diretório src contém os arquivos fontes que deverão ser editados para implementar o projeto. O diretório quartus/ contém o projeto que possibilitará compilar os módulos e testar em hardware.

Entendendo o projeto

A pasta do projeto C no repositório Z01 possui a seguinte estrutura :

```
/C-Logica-combinacional
    /Quartus
    /scripts
        testeLogicaCombinacional.py
    /src
        /rtl
            *.vhd
    /tests
        /tst
            *.vhd
```

1. Quartus : Projeto Quartus que faz uso dos arquivos VHDL localizados em src/rtl/*.vhd

2. scripts : Scripts em python que automatizam a execução dos testes
3. src/rtl/*.vhd : Arquivos VHDL que serão implementado pelo grupo
4. tests/tst/*.vhd : Arquivos VHDL que realizam teste lógico nos arquivos do rtl

Executando o script de teste

Abra o terminal na pasta *C-Logica-Combinacional/script* e execute o script python localizado nessa pasta :

```
$ python testeLogicaCombinacional.py
```

O mesmo irá executar a compilação dos arquivos src/rtl/*.vhd e executar os testes unitários em cada um desses módulos. Como os módulos não estão implementados o resultado deve ser :

```
==== Summary ==================================
pass lib.tb_nand.all            (3.1 seconds)
fail lib.tb_dmux4way.all        (3.3 seconds)
fail lib.tb_or8way.all          (3.3 seconds)
fail lib.tb_dmux2way.all        (3.3 seconds)
fail lib.tb_mux2way.all         (0.5 seconds)
fail lib.tb_dmux8way.all        (1.7 seconds)
fail lib.tb_mux4way.all         (1.7 seconds)
fail lib.tb_barrelshifter16.all (1.7 seconds)
fail lib.tb_or16.all            (1.8 seconds)
fail lib.tb_mux4way16.all       (1.4 seconds)
fail lib.tb_mux8way16.all       (1.6 seconds)
fail lib.tb_not16.all           (1.5 seconds)
fail lib.tb_nor8way.all         (1.5 seconds)
fail lib.tb_and16.all           (1.9 seconds)
fail lib.tb_barrelshifter8.all  (1.7 seconds)
fail lib.tb_mux8way.all         (1.8 seconds)
fail lib.tb_mux16.all           (1.8 seconds)
================================================
pass 1 of 17
fail 16 of 17
================================================
Total time was 33.8 seconds
Elapsed time was 8.7 seconds
================================================
Some failed!
➜  script git:(master) ✗
```

Figure 1: Script teste

Esse comando executa um teste unitário em cada um dos módulos, verificando se sua implementação está correta. O resultado é exibido na tela como : **pass** ou **fail**.

O que deve ser feito :

Esses arquivos estão localizados em C-Logica-Combinacional/src/rtl/ >

Utilize um editor de texto/código de sua preferência para a implementação, valide rodando o script de testes.

Deve-se implementar os seguintes circuitos combinacionais :

- AND 16 bits
  - **Arquivo** : And16.vhd
  - **Descrição** : And bit a bit entre duas palavras de 16 bits.
- OR de 16 bits
  - **Arquivo** : Or16.vhd
  - **Descrição** : OR bit a bit entre duas palavras de 16 bits.
- NOT de 16 bits
  - **Arquivo** : Not16.vhd
  - **Descrição** : NOT bit a bit entre duas palavras de 16 bits.
- NOR 8 Way
  - **Arquivo** : Nor8Way.vhd
  - **Descrição** : NOR entre 8 bits, resulta em uma única saída
- OR 8 Way
  - **Arquivo** : Or8Way.vhd
  - **Descrição** : OR entre 8 bits, resulta em uma única saída
- Barrel Shifter 8 bits
  - **Arquivo** : BarrelShifter8.vhd
  - **Descrição** : Shifta um vetor de 8 bits para a direita e para esquerda.
- Barrel Shifter 16 bits
  - **Arquivo** : BarrelShifter16.vhd
  - **Descrição** : Shifta um vetor de 16 bits para a direita e para esquerda.
- Demultiplexiador de 2 saídas
  - **Arquivo** : DMux2Way.vhd
  - **Descrição** : Demultiplexa uma entrada binária em duas saídas.
- Demultiplexiador de 4 saídas
  - **Arquivo** : DMux4Way.vhd
  - **Descrição** : Demultiplexa uma entrada binária em quatro saídas.
- Demultiplexiador de 8 saídas
  - **Arquivo** : DMux8Way.v8d
  - **Descrição** : Demultiplexa uma entrada binária em oito saídas.
- Multiplexador de duas entradas de 16 bits
  - **Arquivo** : Mux16.vhd
  - **Descrição** : Multiplexa duas entradas de 16 bits para uma de 16 bits.
- Multiplexador 2 entradas de um bit cada
  - **Arquivo** : Mux2Way.vhd
  - **Descrição** : Multiplexa 2 entradas binárias em uma saída binária
- Multiplexador 4 entradas de um bit cada
  - **Arquivo** : Mux4Way.vhd
  - **Descrição** : Multiplexa 4 entradas binárias em uma saída binária
- Multiplexador 8 entradas de um bit cada
  - **Arquivo** : Mux8Way.vhd
  - **Descrição** : Multiplexa 8 entradas binárias em uma saída binária
- Multiplexador 16 entradas de um bit cada

  - **Arquivo** : Mux16Way.vhd
  - **Descrição** : Multiplexa 16 entradas binárias em uma saída binária
- Multiplexador 4 entradas de 16 bits cada
  - **Arquivo** : Mux4Way16.vhd
  - **Descrição** : Multiplexa 4 entradas de 16 bits cada em uma saída de 16 bits.
- Multiplexador 8 entradas de 16 bits cada
  - **Arquivo** : Mux8Way16.vhd
  - **Descrição** : Multiplexa 8 entradas de 16 bits cada em uma saída de 16 bits.

DICAS

Existem diversos locais onde podem tirar dúvida de VHDL, por exemplo

- https://courseware.ee.calpoly.edu/cpe-169/Misc_stuff/cheat_sheet.pdf
- http://esd.cs.ucr.edu/labs/tutorial/
- https://www.ics.uci.edu/~jmoorkan/vhdlref/vhdl_golden_reference_guide.pdf

NAND.vhd

A seguir a implementação da And16.vhd

```
Library ieee;
use ieee.std_logic_1164.all;

entity nand_z01 is
   port(
          a : in  std_logic;
          b : in  std_logic;
          q : out std_logic
   );
end nand_z01;

architecture rtl of nand_z01 is
begin
   q <=  not (a and b);
end rtl;
```

Testando em HW

Para testar os módulos em hardware, deve abrir o projeto (C-Logica-Combinacional/Quartus) e importe os módulos que irá utilizar (port map) utilizando-os em seu projeto.

Rubricas para avaliação de projetos

Cada integrante do grupo irá receber duas notas : Uma referente ao desenvolvimento total do projeto (Projeto) e outra referente a sua participação individual no grupo (que depende do seu papel).

Projeto

Conceito	Descritivo
I	- Menos que 12 módulos implementados e funcionando
D	- Menos que 15 módulos implementados e funcionando
C	- Ter criado o project no github - O travis configurado e funcionando - Ao menos 15 módulos implementados e funcionando (teste ok) - Implementando a equação do grupo no Quartus sem usar port map
B	- Todos os módulos implementados e funcionando (teste ok) - Implementando a equação do grupo com port map (usando os componentes /src/rtl/)
A	- Projeto bem documentando e comentado

Desenvolvedor

Conceito	Descritivo
I	- Se comprometeu a fazer algum desenvolvimento (kanban) e não fez.
D	- Criou o branch, mas não fez o desenvolvimento completo.
C	- Desenvolveu as rotinas para passarem nos testes ! - Acompanhar Kanban board (github project). Ex: Puxou tarefas, etc. - Respeitar a arquitetura acordada. (vale negociação com colegas para mudanças, contando que estejam documentadas) - Submeter alterações por Pull requests.
B	- Criar um Branch com commits autocontidos (ou seja, focados no problema e não interferindo em outras parte do projeto). Ex: atualizar com Pull, fazer iterações… - Associa um pull-request ao issues

Conceito	Descritivo
A	- Desenvolver código modularizado, comentado e claro. Usando as ferramentas e técnicas mais apropriadas. Ex: Organizado em Classes e Objetos com baixo nível de acoplamento. - Faz comentários pertinentes no pull-request, facilitando o trabalho do facilitador

Facilitador

Conceito	Descritivo
I	- Não acompanha projeto, deixa colegas sozinho - O relatório das atividades não é condizente com o real desempenho dos integrantes (analisado via git)
D	- Não verificar se o pull request é consistente (não altera arquivos de outros). - O relatório das atividades não é claro e não descrevendo cada integrante - Não consegue demonstrar o projeto em operação ou explicar possíveis problemas
C	- Atualizar o repositório pelo Fork. - Dar manutenção Kanban board (GitHub project). Ex: criar cards - Acompanha e dar feedback mínimo para membros do time. - Aceitar os pull-requests - Fez o relatório das atividades descrevendo o papel de cada integrante com clareza
B	- Resolver conflitos de Merge dos Pull Requests – Dar feedback usando as ferramentas dos sistemas (Git e Slack e Github). Ex: Comentário e pull requests do Git. - Reforçar a arquitetura do sistema. - Fez uma boa distribuição homogênea das tarefas
A	- Agregou informação relevante para o grupo melhor desenvolver o projeto. Ex: ranqueou atividades no Kanban, documentação extra.