Insper

Elementos de Sistema - Projeto D - Unidade Lógica Aritmética

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

Março - 2018

> **!** Prazo original : Quarta Feira - 20/3/2018
> Atrasado até : Quarta Feira - 27/3/2018

Descrição

Neste projeto você terá de desenvolver os componentes para a implementação de uma unidade lógica aritmética completa de 16 bit (proposta pelo livro texto). Os alunos deverão subdividir a ALU em módulos e duplas de alunos desenvolverão cada parte. O facilitador escolhido será responsável pela completude e consistência do branch master.

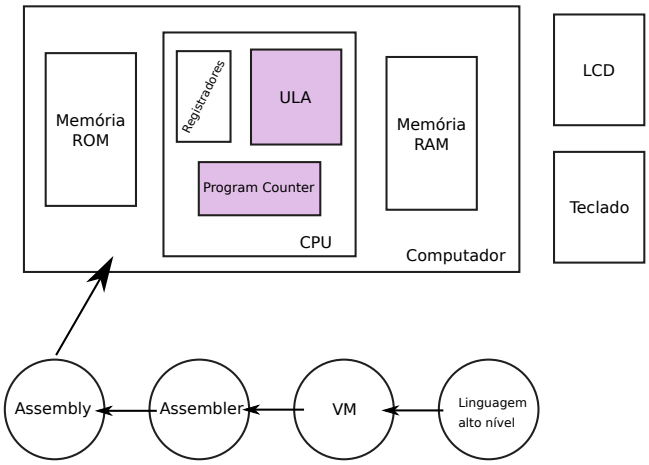

Figure 1: ULA

Integrantes

Este projeto é para ser realizado por todos os integrantes do grupo. Tarefas devem ser criadas no Kambam 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.

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 :

```
/D-ULA
    /Quartus
    /scripts
        testeULA.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 automatiza 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 *D-UnidadeLogicaAritmetica/script* e execute o script python localizado nessa pasta :

```
$ python testeULA.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 deverá ser falho.

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 :

Nesse projeto, deve-se além de implementar os módulos, deve-se gerar uma imagem com a forma de onda desse módulo. Agora somente os módulos que são selecionados para testar serão testados, para cada nova implementação criar um novo branch e descomentar somente o módulo que está implementando.

Note que é possível reaproveitar (via port map) os módulos do projeto anterior (C), os mesmos já são incluídos automaticamente (pelo script) na compilação

desses módulos.

Módulos

Deve-se implementar os seguintes circuitos combinacionais :

- HalfAdder
  - **Arquivo** : HalfAdder.vhd
  - **Descrição** : Adiciona dois bits que resulta em uma soma e carry
- FullAdder
  - **Arquivo** : FullAdder.vhd
  - **Descrição** : Adiciona três bits que resulta em uma soma e carry
- Add16
  - **Arquivo** : Add16.vhd
  - **Descrição** : Adiciona dois vetores de 16 bits resultando em um outro vetor de 16 bits (sem carry)
- Inc16
  - **Arquivo** : Inc16.vhd
  - **Descrição** : Adiciona '1' a um vetor de 16 bits resultando em um sinal de 16 bits (sem carry).
- Inversor16
  - **Arquivo** : Inversor16.vhd
  - **Descrição** : Inverte um vetor de entrada quando o bit de controle **z** for igual a '1', e não modifica o vetor de entrada caso contrário, o resultado é um novo vetor de 16 bits.
- Zerador16
  - **Arquivo** : Zerador16.vhd
  - **Descrição** : Zera um vetor de entrada quando o bit de controle **z** for igual a '1', e não modifica o vetor de entrada caso contrário, o resultado é um novo vetor de 16 bits.
- Comparador16
  - **Arquivo** : Comparador16.vhd
  - **Descrição** : Compara se um vetor de 16 bits é igual a zero se for, faz com que o sinal **zr** seja igual a '1' e '0' caso contrário. Se o sinal for negativo faz com que **ng** seja igual a '1', caso contrário (positivo) '0'.
- ALU
  - **Arquivo** : ALU.vhd
  - **Descrição** : A entidade que faz o mapeamento de todas as demais, interligando os blocos (zerador, comparador, inversor, Add ....) em um único bloco.

Para implementar a ALU será necessário usar os bócos desenvolvidos nesse

projeto mais os blocos desenvolvidos no projeto anterior : And16, Mux16. O script de compilação e teste já faz a inclusão deles. A arquitetura da ULA é a :

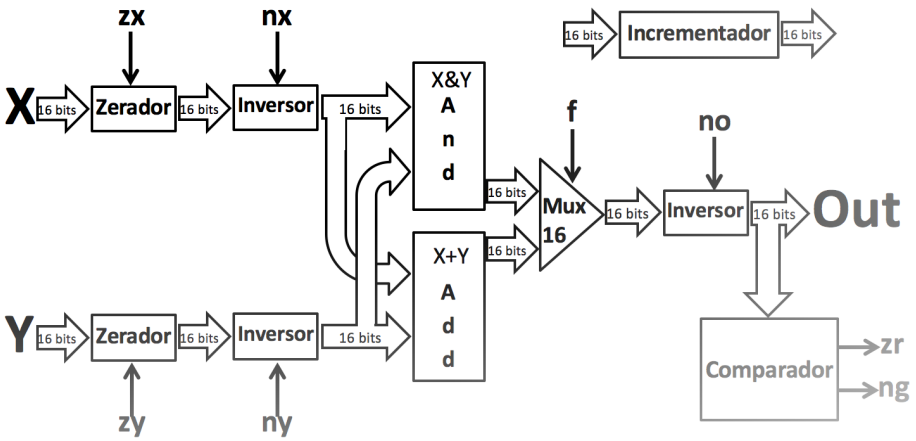

Figure 2: ULA

Forma de onda

Para cada teste realizado, deve-se carregar a interface gráfica e tirar um print da forma de onda do módulo com os testes aplicados a ele (D-Ferramental.pdf), essa imagem deve ser salva na mesma pasta dos arquivos vhdl (src/rtl/) e com o mesmo nome dos módulos. A pasta no final do projeto deve possuir os seguintes arquivos :

```
/src/rtl/
        Add16.vhd
        Add16.png
        ALU.vhd
        ALU.png
        Comparador16.vhd
        Comparador16.png
        FullAdder.vhd
        FullAdder.png
        HalfAdder.vhd
        HalfAdder.png
        Inc16.vhd
        Inc16.png
        Inversor16.vhd
        Inversor16.png
        Zerador16.vhd
```

`Zerador16.png`

Testando em HW

Para testar os módulos em hardware, deve abrir o projeto (D-UnidadeLogicaAritmetica/Quartus) , esse projeto já inclui todos os módulos dessa entrega e também os módulos da entrega passada. O arquivo localizado em /rtl/toplevel.vhd já faz o mapeamento dos pinos da FPGA para os pinos da ULA, assim como o handout da aula 7, para testar no hardware basta compilar e programar a FPGA.

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	
I	- Não implementou os módulos Add16, ULA, Comparador, FullAdder, HalfAdder, Inc16, Inversosr, Zerador
D	- Implementou todos os módulos menos a ULA
C	- O travis configurado e funcionando - Todos os testes ok - ULA funcionando - Todos os módulos sendo testados no Travis - Possui a forma de onda de todos os módulos (.png)
B	- Gravou a ULA na FPGA e fez um vídeo demonstrando o seu funcionamento
A	- Usou port map para reaproveitar outros módulos, por exemplo: - usou o full-adder para criar o add16 , usou na ULA o mux do projeto C

Desenvolvedor

Conceito	
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.

Conceito	
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 - Adiciona a forma de onda gerada nos testes ao pull-request
A	- Desenvolver código modularizado, comentado e claro. Usando as ferramentas e técnicas mais apropriadas. Ex: port map

Facilitador

Conceito	
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.