

PCS3225 - Sistemas Digitais II

Projeto 3 - Unidades Funcionais em VHDL

Deadline: 7/11/2022

O objetivo deste trabalho é exercitar o projeto e descrição de unidades funcionais em VHDL, que serão usadas como componentes internos do PoliLEGv8, o processador que desenvolveremos nesta disciplina.

Você descreverá a ULA completa.

Introdução

Um processador pode ser visto como um projeto que utiliza o paradigma de unidade de controle e fluxo de dados. A unidade de controle é responsável pelo ciclo de busca, decodificação, execução e gravação dos dados, ciclo este que rege o funcionamento de um computador de uso geral pela sua característica programável. Já o fluxo de dados é composto por elementos de memória (e.g. banco de registradores), componentes de controle de fluxo (quase sempre combinatórios) e **unidades funcionais**.

O cálculo computacional é realizado nas unidades funcionais, portanto não é exagero afirmar que são os componentes principais de um processador. É possível construir máquinas com um ou até mesmo sem registradores, mas uma máquina sem uma unidade funcional simplesmente não realiza computação alguma.

Como as unidades funcionais são muito comuns, é costume juntar as principais funções computacionais em uma única unidade, que chamamos de ULA (Unidade Lógica e Aritmética). Como o próprio nome diz, uma ULA reúne em um único componente operações lógicas (e.g. AND, OR, XOR, etc.) e aritméticas (adição, subtração, etc.), além de operações de comparação (menor, maior, igual, etc.).

ALU, Arithmetic and Logic Unit.

Atividades

A ULA do PoliLEG, como várias outras implementações, agrupa algumas unidades funcionais. No caso do PoliLEG monociclo, a ULA é capaz de realizar as operações aritméticas adição e subtração, as lógicas AND, OR e NOR, e ainda uma operação de comparação. A Tabela 1 sumariza as 6 operações capazes de serem realizadas nesta ULA.

Dica: Veja a construção da ULA no Apêndice A.5 da Referência [1].

OP	Operação	Descrição
0000	AND	$A \& B$, bit a bit
0001	OR	$A B$, bit a bit
0010	soma	$A + B$
0110	subtrai	$A - B$
0111	Set on Less Than (SLT)	$A < B$
1100	NOR	$\overline{A B}$, bit a bit

Tabela 1: Operações que podem ser realizadas pela ULA. O campo OP pode ser interpretado como $(Op_3Op_2Op_1Op_0)$, onde: Op_3 inverte A, Op_2 inverte B, e Op_1Op_0 escolhe a unidade funcional entre: 00:AND, 01:OR, 10:ADD (somador), 11:SLT.

Sabemos então que a ULA possui efetivamente três unidades funcionais, a saber AND, OR e um somador, além da capacidade de

A operação SLT usa o somador e fios.

inverter qualquer entrada independentemente. As operações AND, OR e adição são realizadas diretamente pelas unidades correspondentes. A subtração pode ser obtida invertendo-se B e entrando-se com 1 no *carry-in*, realizando a operação de soma com o complemento de base do número, o que equivale à subtração. A operação NOR é realizada pela unidade de AND, invertendo-se as entradas. Já a operação Set on Less Than diferencia-se das demais pois o resultado é 1 (decimal) se $A < B$, assumindo-se que não há overflow em $A - B$, ou 0 (decimal) caso contrário.

```
entity alu is
  generic (
    size : natural := 10 — bit size
  );
  port (
    A, B : in  bit_vector(size-1 downto 0); — inputs
    F : out bit_vector(size-1 downto 0); — output
    S : in  bit_vector(3 downto 0); — op selection
    Z : out bit; — zero flag
    Ov : out bit; — overflow flag
    Co : out bit — carry out
  );
end entity alu;
```

Listagem 1: Entidade para a ULA

Os sinais de estado são as saídas que indicam que o resultado é zero ou que o resultado é *overflow*. Além dos sinais de estados, ainda há uma saída de *carry* e a saída com o resultado F. A Figura 1 mostra o símbolo para a ULA e a Listagem 1 a entidade correspondente em VHDL.

Assuma que *carry-in* deve ser 1 sempre que B for invertido.

Use o teorema de DeMorgan para entender o NOR.

Na realidade, SLT nos dá a flag N de $A - B$, que indica se $A < B$ quando não há overflow.

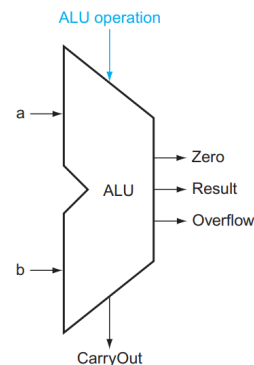


Figura 1: Diagrama da ULA.

Note que não há entrada de *carry*.

P3A1 (4 pontos) Implemente um componente em VHDL correspondente a uma ULA de 1 bit, respeitando a entidade na Listagem 2.

Projeto 3, Atividade 1

```
entity alu1bit is
  port (
    a, b, less, cin: in bit;
    result, cout, set, overflow: out bit;
    ainvert, binvert: in bit;
    operation: in bit_vector(1 downto 0)
  );
end entity;
```

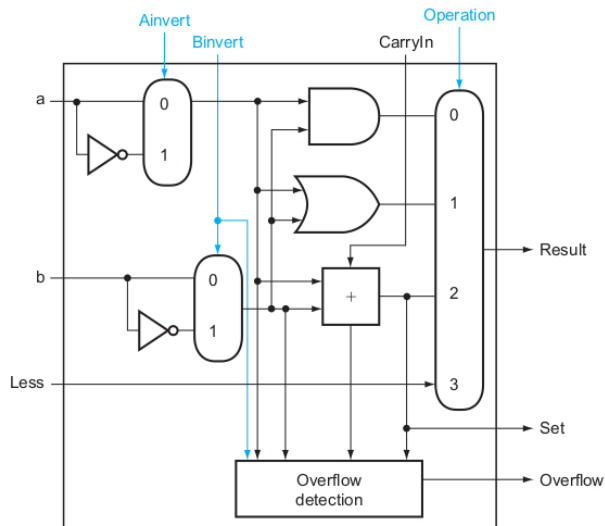
Listagem 2: Entidade para a ULA de 1 bit.

```
entity fulladder is
  port (
    a, b, cin: in bit;
    s, cout: out bit
  );
end entity;
```

Listagem 3: Entidade do somador completo.

A ULA de 1-bit possui as seguintes características: opera sempre sobre as entradas a e b, ou sobre suas versões negadas caso ainvert ou binvert estejam ativos, respectivamente. As funções são AND, OR, ADD ou SLT, selecionáveis através de um multiplexador que decide qual resultado será colocado na saída result (F), na ordem mostrada, onde AND equivale a operation=00 e SLT a operation=11.

As operações realizadas nas unidades AND e OR são *bitwise*. A adição leva em consideração o *carry-in* e pode gerar um *carry-out*. A saída set é uma cópia do resultado do somador, independente da seleção do multiplexador de saída. A saída overflow é alta quando houver *overflow* no somador, independente da seleção do multiplexador de saída e considerando que este módulo formará uma ULA de múltiplos bits. Por último, caso o multiplexador selecione a função SLT, a entrada less é copiada para a saída. A Figura 2 mostra a estrutura de uma possível implementação desta ULA de 1-bit.



Bitwise: bit a bit.

A saída *overflow* só faz sentido e só será usada no bit mais significativo.

Figura 2: Diagrama da ULA de 1 bit.

Você pode utilizar o somador completo, fornecido no edisciplinas, cuja entidade está na Listagem 3.

P3A2 (6 pontos) Implemente um componente em VHDL correspondente a ULA completa, que respeite a entidade na Listagem 1. Note que esta ULA possui um parâmetro genérico e pode ser instanciada com qualquer número de bits maior que 1.

Você pode assumir que há uma ULA de 1-bit correta e funcional, mesmo que não tenha entregue a Atividade 1. Sua utilização para montar a ULA genérica é opcional.

Instruções para Entrega

Há um link específico no e-Disciplinas para a submissão de cada atividade deste projeto. Acesse-o somente quando estiver confortável para enviar sua solução. Em cada atividade, você pode enviar apenas um único arquivo codificado em UTF-8. O nome do arquivo não importa, mas sim a descrição VHDL que está dentro. A entidade da síntese (implementação) que você submeterá precisa estar, obrigatoriamente, de acordo com o que foi definido/especificado no enunciado e deve ser idêntica na sua solução ou o juiz não irá processar seu arquivo.

Projeto 3, Atividade 2

Qualquer editor de código moderno suporta UTF-8 (e.g. Atom, Sublime, Notepad++, etc)

Na atividade 1, **você não deve incluir o fulladder** (somador completo) no seu arquivo, e, na atividade 2, **não deve constar a ULA de 1 bit**. Ambos componentes já estarão no juiz.

O juiz corrigirá imediatamente sua submissão e retornará com a nota. Caso não esteja satisfeito com a nota, você pode enviar novamente e somente a melhor nota para aquela atividade será válida. A nota para este projeto é composta pela soma ponderada das notas dadas pelo juiz para cada atividade. Faça seu *testbench* e utilize um simulador de VHDL para validar sua solução antes de postá-la para o juiz.

Não use a biblioteca `std_logic_1164`, a `textio` e qualquer outra biblioteca não padronizada, para minimizar possível fonte de problemas. Também se certifique que não imprime nada na saída da simulação.

Atenção: não atualize a página de envio e não envie a partir de conexões instáveis (e.g. móveis) para evitar que seu arquivo chegue corrompido no juiz.

A quantidade de submissões para estes problemas foi limitada a 5 por problema.

Referências

- [1] D. Patterson and J. Hennessy. *Computer Organization and Design ARM Edition: The Hardware Software Interface*. The Morgan Kaufmann Series in Computer Architecture and Design. Elsevier Science, 2016.