

PCS3225 - Sistemas Digitais II

Trabalho 3 - Unidades Funcionais em VHDL

Bruno de Carvalho Albertini

11/10/2019

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

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 operações capazes de serem realizadas nesta ULA.

| OP | Unidade | Descrição |
|------|-----------|--------------------------------|
| 0000 | AND | $A \& B$, bit a bit |
| 0001 | OR | $A B$, bit a bit |
| 0010 | adição | $A + B$ |
| 0110 | subtração | $A - B$ |
| 0111 | SLT | $A < B$ |
| 1100 | AND | $\overline{A B}$, 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 função entre: 00:AND, 01:OR, 10:+, 11:less.

Sabemos então que a ULA possui efetivamente três unidades funcionais, a saber AND, OR e um somador, além da capacidade de 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 a subtração. A operação NOR é realizada pela unidade de AND, invertendo-se as entradas. Já a operação SLT diferencia-se das demais pois o resultado é 1 (decimal) se $A < B$ 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. A Figura 1 mostra o símbolo para a ULA e a Listagem 1 a entidade correspondente em VHDL.

Todos os números são interpretados como complemento de 2.

Use o teorema de DeMorgan para entender o NOR.

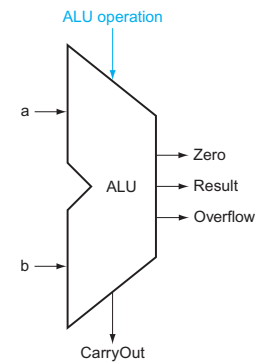


Figura 1: Diagrama da ULA.

Note que não há entrada de *carry*.

T3A1 (2 pontos) Implemente um componente em VHDL correspondente a uma ULA de 1 bit, respeitando a entidade na Listagem 2.

Trabalho 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 operações são AND, OR, ADD ou SLT, selecionáveis através de um multiplexador que decide qual resultado será colocado na saída *result*, na ordem mostrada,

onde AND equivale a $\text{operation}=00$ e SLT a $\text{operation}=11$.

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 é módulo formará uma ULA de múltiplos bits. Por último, caso o multiplexador selecione a operação SLT, a entrada less é copiada para a saída.

É opcional montar a ULA de 1-bit com ou sem utilização do somador completo cuja entidade está na Listagem 3. Caso o utilize, não é necessário incluí-lo na sua solução.

Bitwise: bit a bit.

A saída *overflow* só faz sentido no MSB.

T3A2 (8 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.

Trabalho 3, Atividade 2

Instruções para Entrega

Você deve acessar <https://pelicano.pcs.usp.br>, logar com o email cadastrado no Júpiter e enviar os seus dois arquivos, um de cada vez. Durante o envio do arquivo, será solicitada a *tag* do problema a ser resolvido, que está abaixo:

| Tarefa | Problema | tag |
|--------|--------------|-------|
| T3A1 | ULA 1-bit | ULA01 |
| T3A2 | ULA genérica | ULA64 |

Note que os *tags* são diferentes.

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

Atenção: não atualize a página de submissões e não envie a partir de conexões instáveis (e.g. móveis).

A quantidade de submissões para estes problemas foi limitada a 10 por problema.