PCS3225 - Sistemas Digitais II
Atividade Formativa 11 - Projeto do Processador PoliLEGv8 em VHDL
Parte 3 - Unidades Funcionais do PoliLEGv8

Glauber de Bona, revisado por Edson Toshimi Midorikawa (2023) e Antonio Vieira da Silva Neto (2024)

Data do Arquivo: 11/11/2024; Prazo da AF11: 23/11/2024

O objetivo desta parte da Atividade Formativa 11 é exercitar o projeto e descrição de unidades funcionais em VHDL, que serão usadas como componentes internos do PoliLEGv8.

Você descreverá a ULA do PoliLEGv8.

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, orquestrando o funcionamento de um computador de uso geral pela sua característica programável. Já o fluxo de dados, por sua vez, é composto por elementos de memória (e.g. banco de registradores), componentes de controle de fluxo (quase sempre combinatórios) e **unidades funcionais**.

Como o cálculo computacional é realizado nas unidades funcionais, não é exagero afirmar que elas representam uma parte fundamental de um processador. É possível construir máquinas com um ou até mesmo nenhum registrador, mas uma máquina sem uma unidade funcional simplesmente não realiza computação alguma.

Como as unidades funcionais são muito comuns, é usual juntar as principais funções computacionais em uma única unidade multifuncional denominada 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.), incluindo operações de comparação (menor, maior, igual, etc).

Em inglês, ALU, *Arithmetic and Logic Unit*.

Conceitos

A ULA do PoliLEGv8 agrupa algumas unidades funcionais. No caso do PoliLEGv8 monociclo, a ULA é capaz de realizar as operações aritméticas de adição e subtração, as operações lógicas AND, OR e NOR e uma operação de comparação. A Tabela 1 resume as seis operações passíveis de execução na ULA.

Dica: Veja o material complementar da Aula 18 (Projeto do Processador PoliLEGv8) para mais detalhes sobre a construção da ULA.

Nota: *Dado o conjunto de instruções do PoliLEGv8, apenas as operações lógicas AND e OR e as operações aritméticas de soma e subtração são diretamente utilizadas pelo programador. A operação "Pass B"é utilizada para detectar se um registrador possui valor zero em desvios condicionais do tipo "CBZ". A operação NOR está disponível para ser utilizada pelo programa-*

OP	Operação	Descrição
`0000`	`AND`	$A\&B$, bit a bit
`0001`	`OR`	$A\|B$, bit a bit
`0010`	Soma	$A + B$
`0110`	Subtração	$A - B$
`0111`	Repasse da Entrada B ("Pass B")	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 ($Op_3Op_2Op_1Op_0$), sendo 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:Pass B`.

dor em extensões futuras do conjunto de instruções do processador.

Sabe-se que a ULA possui, na prática, 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 é obtida invertendo-se a entrada `B`, entrando-se com `1` no *carry-in*, e realizando a soma, com o complemento de base do número. A operação `NOR` é realizada pela unidade `AND`, invertendo-se as entradas. Já a operação `Pass B` (Repasse da Enrada B) consiste em replicar a entrada B na saída da ULA..

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

Use o teorema de DeMorgan para entender o `NOR`.

Na prática, `Pass B` é utilizada após a integração de várias ULAs para verificar alguma condição externa sobre a entrada B, - por exemplo, se ela é zero nos desvios condicionais do tipo "CBZ".

A Figura 1 mostra o símbolo para a ULA e a Listagem 1 corresponde à entidade da ULA do PoliLEGv8 em VHDL. Note que a ULA recebe como entradas dois dados com 64 bits de largura, denominados A e B, e que ela possui um sinal de controle de 4 bits denominado S, que serve para determinar a operação a ser executada. As saídas da ULA são o resultado da operação executada, denominada F e que também possui 64 bits de largura, e três *flags*: Z para indicar se o resultado da operação é zero, Ov para indicar se houver *overflow* e Co (*carry out*) para indicar se houve produção de vai-um ao final do processamento.

No material didático (regular e complementar), o sinal de controle S é denominado *ALU control.*

Note que não há entrada de *carry*.

Os três *flags* são ativos em nível alto, tal como todo sinal do PoliLEGv8. Por exemplo, se o resultado de uma operação for zero, a saída Z da ULA será igual a 1.

```
entity ula is
  port (
    A  : in  bit_vector(63 downto 0); -- entrada A
    B  : in  bit_vector(63 downto 0); -- entrada B
    S  : in  bit_vector(3 downto 0);  -- seleciona operacao
    F  : out bit_vector(63 downto 0); -- saida
    Z  : out bit;                     -- flag zero
    Ov : out bit;                     -- flag overflow
    Co : out bit                      -- flag carry out
  );
end entity alu;
```

Listagem 1: Entidade para a ULA

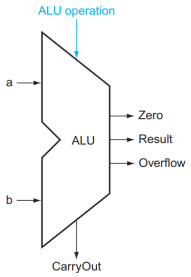

Figura 1: Diagrama da ULA.

ULA do PoliLEGv8

O circuito da ULA do PoliLEGv8 pode ser implementado como uma **composição de 64 ULAs elementares** de 1 bit, chamadas `ula1bit`. A figura 2 mostra a estrutura interna da ULA de 1 bit. Pode-se notar que é composta pelas portas AND e OR e um somador completo de 1 bit, além de inversores e de um multiplexador. A lógica de deteção

Use os comandos `for-generate` e `if-generate` do VHDL.

de *overflow* gera a saída *overflow* levando em consideração os valores das entradas e o resultado da soma.

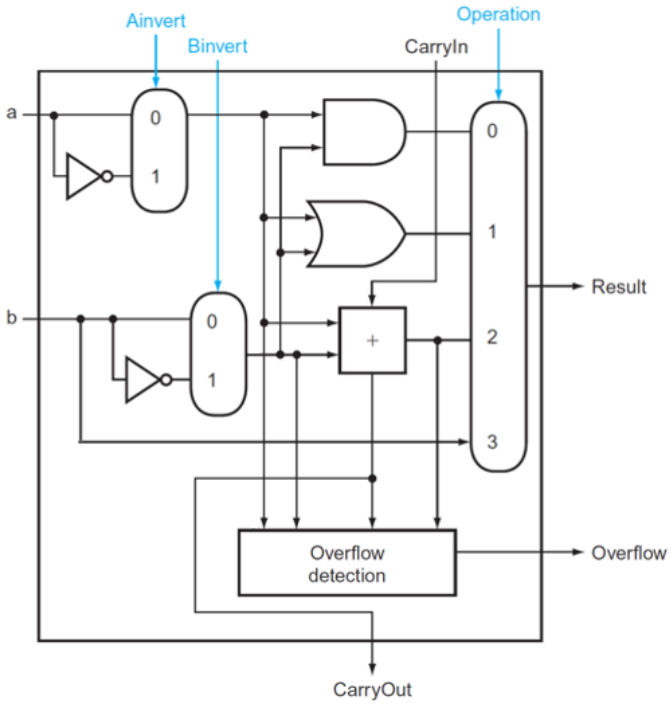

Figura 2: Diagrama da ULA de 1 bit.

ULA de 1 bit

A ULA de 1 bit possui as seguintes características: opera sempre sobre as entradas `a` e *b*, ou sobre suas versões negadas caso as entradas `ainvert` ou `binvert` estejam, respectivamente, ativas. As funções da ULA de 1 bit são `AND`, `OR`, `ADD` ou `Pass B`, e devem ser escolhidas por meio de um multiplexador que decide qual resultado será colocado na saída `result` (*F*) na ordem mostrada. Dessa forma, `AND` é a operação da saída quando `operation=00`; `OR` é a operação da saída quando `operation=01`; `ADD` é a operação da saída quando `operation=10`; e `Pass B` é a operação da saída quando `operation=11`.

As operações realizadas nas unidades `AND` e `OR` são bit a bit (*bitwise*). A adição leva em consideração o *carry-in* e pode gerar um *carry-out*. A saída `overflow` é alta quando houver *overflow* no somador, independente da seleção do multiplexador de saída e considerando que esse módulo formará uma ULA de múltiplos bits. Por último, caso o multiplexador selecione a função `Pass B`, a entrada `B` é copiada para a saída.

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

A figura 3 mostra a estrutura interna da ULA mostrando como as unidades elementares formam a composição final. Repare também que a saída `Zero` é gerada por uma porta NOR. Estude essa figura e procure encontrar o padrão repetitivo de conexão de um componente a outro.

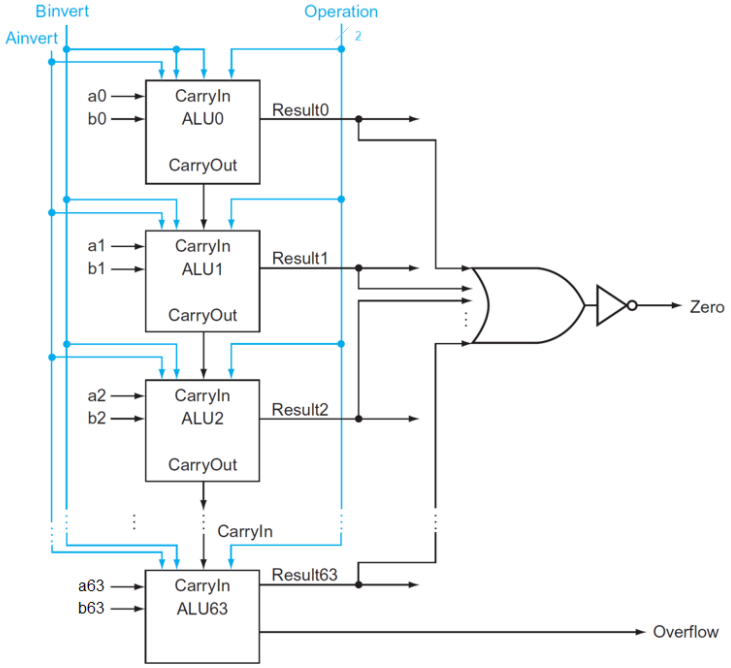

Figura 3: Diagrama da estrutura interna da ULA do PoliLEGv8.

Atividades

AF11-P3E1 Implemente um componente em VHDL correspondente a uma ULA de 1 bit que seja aderente à entidade da Listagem 2.

Atividade Formativa 11 Parte 3, Exercício 1

```
entity ula1bit is
  port (
    a         : in  bit;
    b         : in  bit;
    cin       : in  bit;
    ainvert   : in  bit;
    binvert   : in  bit;
    operation : in  bit_vector(1 downto 0);
    result    : out bit;
    cout      : out bit;
    overflow  : out bit
  );
end entity;
```

Listagem 2: Entidade para a ULA de 1 bit.

Você pode utilizar o somador completo (`fulladder.vhd`), fornecido no e-Disciplinas, cuja entidade está na Listagem 3.

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

Listagem 3: Entidade do somador completo.

Elabore um *testbench* para fazer a verificação lógica da entidade da Atividade. Defina os **casos de teste** necessários no processo de teste e verificação do projeto.

AF11-P3E2 Implemente um componente em VHDL correspondente à ULA de 64 bits do PoliLEGv8, que respeite a entidade na Listagem 1. Utilize a ULA de 1 bit do exercício **AF11-P3E1** como base para projetar a ULA de 64 bits.

Atividade Formativa 11 Parte 3 - Exercício 2

Elabore um *testbench* para fazer a verificação lógica da ULA do PoliLEGv8. Defina os **casos de teste** necessários no processo de teste e verificação do projeto.

Instruções para os Grupos

Como boa prática de projeto de Engenharia, faça seus *testbenches* e utilize o GHDL/GtkWave ou *EDA Playground* (selecionando o GHDL como simulador) para validar suas soluções.

Use as técnicas apresentadas no módulo de **Verificação de Projetos** para acrescentar formas de verificação de funcionamento para validar o projetos das diversas atividades do projeto.
Atenção: É vedado o uso da biblioteca `ieee.std_logic_1164`, ou qualquer outra biblioteca não padronizada, no projeto do PoliLEGv8.