Universidade de Brasília

Trabalho 4: Projeto e Simulação de uma ULA em VHDL

Universidade de Brasília
Departamento de Ciência da Computação
Organização e Arquitetura de Computadores
Evelyn Soares Pereira
17/0102785
soares.evelynp@gmail.com

1. DESCRIÇÃO DO TRABALHO

Este trabalho consiste na implementação de uma ULA em VHDL no ambiente EDA Playground. A Unidade Lógico-Aritmética (ULA) é um componente em um processador que realiza operações lógicas e aritméticas em dados. Ela é responsável por executar as operações fundamentais necessárias para o processamento de informações em um sistema digital. As operações típicas incluem adição, subtração, operações lógicas (AND, OR, XOR), comparações (como menor que e igual), e deslocamentos de bits.

O código VHDL deste trabalho representa consiste nas declarações e bibliotecas, após isso, declara a entidade ulaRV que representa a ULA. Possui parâmetros genéricos, como o tamanho dos operandos (WSIZE). Tem portas de entrada (opcode, A, B) e saídas (Z, zero). Um único processo sensível às mudanças em A, B, e opcode. Inicializa zero como '0'. Após isso faz o case statement (estrutura de seleção) para determinar a operação com base no valor de opcode.

Para ADD ("0000") e SUB ("0001"), realiza adição e subtração aritmética com sinal, respectivamente. AND ("0010") e OR ("0011") e XOR ("0100"), realiza operações lógicas bit a bit (AND, OR, XOR) nos operandos A e B. SLL ("0101") e SRL ("0110") e SRA ("0111") realiza operações de shift (deslocamento) à esquerda, à direita lógica e à direita aritmética. SLT ("1000") e SLTU ("1001") e SGE ("1010") e SGEU ("1011") realiza comparações (menor que, menor ou igual, maior que, maior ou igual) com e sem sinal.Por fim, SEQ ("1100") e SNE ("1101") realiza comparações de igualdade e desigualdade. Define Z como 'Z' (alta impedância) se o valor de opcode não corresponder a nenhuma operação conhecida. Os valores dos opcodes forma definidos no roteiro, conforme a figura 1.

2. TESTES REALIZADOS

Para verificar o funcionamento da ULA, foi feita a declaração de sinais, a declaração do componente ULA, ulaRV, declarado utilizando a declaração component, e suas portas são associadas aos sinais declarados. Depois, a instanciação da ULA dando começo ao processo de testes.

O testbench realiza uma série de operações (ADD, SUB, AND, OR, XOR, SLL, SRL, SRA, SLT, SLTU, SGE, SGEU, SEQ, SNE) com diferentes valores de operandos.

ADD: 'A + B', onde 'A = 2' e 'B = 2'. O resultado ('Z') esperado é '4'.

SUB: 'A - B', onde 'A = 2' e 'B = 5'. O resultado ('Z') esperado é '-3'.

AND: 'A AND B', onde 'A = 15' e 'B = 51' (em binário). O resultado ('Z') esperado é '3'.

OR: 'A OR B', onde 'A = 15' e 'B = 240' (em binário). O resultado ('Z') esperado é '255'.

XOR: - 'A XOR B', onde 'A = 15' e 'B = 5'. O resultado ('Z') esperado é '10'.

SLL: - Deslocamento lógico para a esquerda de 'A' por 'B' bits, onde 'A = 1' e 'B = 3'. O resultado ('Z') esperado é '8'.

SRL: - Deslocamento lógico para a direita de 'A' por 'B' bits, onde 'A = 65535' e 'B = 2'. O resultado ('Z') esperado é '16383'.

SRA: - Deslocamento aritmético para a direita de 'A' por 'B' bits, onde 'A = -32768' e 'B = 1'. O resultado ('Z') esperado é '-16384'.

SLT: - Teste se 'A < B', onde 'A = -2' e 'B = 15'. O resultado ('Z') esperado é '1' (true).

SLTU: - Teste se 'A < B' sem sinal, onde 'A = 51' e 'B = 195'. O resultado ('Z') esperado é '1' (true).

SGE: - Teste se 'A >= B', onde 'A = 1' e 'B = 1'. O resultado ('Z') esperado é '1' (true).

SGEU: - Teste se 'A >= B' sem sinal, onde 'A = 51' e 'B = 195'. O resultado ('Z') esperado é '0' (false).

SEQ: - Teste se 'A = B', onde 'A = 15' e 'B = 15'. O resultado ('Z') esperado é '1' (true).

SNE: - Teste se 'A B', onde 'A = -1' e 'B = 2'. O resultado ('Z') esperado é '1' (true).

Operação	Significado	OpCode
ADD A, B	Z recebe a soma das entradas A, B	0000
SUB A, B	Z recebe A - B	0001
AND A, B	Z recebe a operação lógica A and B, bit a bit	0010
OR A, B	Z recebe a operação lógica A or B, bit a bit	0011
XOR A, B	Z recebe a operação lógica A xor B, bit a bit	0100
SLL A, B	Z recebe a entrada A deslocada B bits à esquerda	0101
SRL A, B	Z recebe a entrada A deslocada B bits à direita sem sinal	0110
SRA A, B	Z recebe a entrada A deslocada B bits à direita com sinal	0111
SLT A, B	Z = 1 se A < B, com sinal	1000
SLTU A, B	Z = 1 se A < B, sem sinal	1001
SGE A, B	Z = 1 se A ≥ B, com sinal	1010
SGEU A, B	Z = 1 se A ≥ B, sem sinal	1011
SEQ A, B	Z = 1 se A == B	1100
SNE A, B	Z = 1 se A != B	1101

Figure 1. Tabela de Opcode com suas respectivas operações.

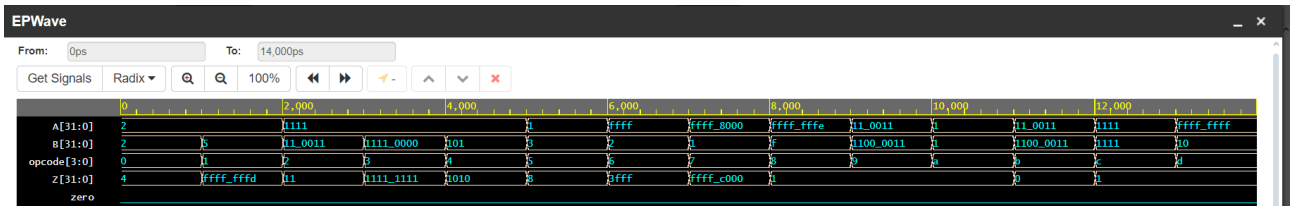

Figure 2. Simulação.

3. SIMULAÇÃO E RESULTADOS

Na comparação com sinal, o valor mais significativo (MSB) de um número é reservado para representar o sinal do número (positivo ou negativo). Assim, ao comparar dois números com sinal, você está levando em consideração a polaridade dos valores. Por exemplo, em um número com sinal de 8 bits, o valor 01111111 é positivo, enquanto o valor 10000000 é negativo. Por outro lado, na comparação sem sinal, todos os bits de um número são tratados como magnitudes positivas, e não há um bit de sinal específico. Isso significa que todos os valores são considerados positivos, e não há distinção entre positivo e negativo. A comparação sem sinal é usada quando você está interessado apenas na magnitude dos números, sem se preocupar com a polaridade.

A detecção de overflow em operações de adição (ADD) e subtração (SUB) é crucial para garantir que os resultados sejam válidos e não ultrapassem os limites representáveis. Para overflow em adição (ADD), com sinal, no caso de adição com sinal, overflow ocorre quando os operandos têm o mesmo sinal e o resultado tem sinal oposto. Isso pode ser detectado observando o carry no bit mais significativo (MSB) e o carry-out do segundo bit mais significativo. Se for sem sinal, ocorre overflow se o carry-out do MSB for 1. Já para overflow em subtração (SUB), na subtração com sinal, overflow ocorre quando os operandos têm sinais opostos e o resultado tem sinal oposto ao subtraindo. Isso pode ser detectado observando o borrow do MSB e o borrow-out do segundo bit mais significativo. Em subtrações sem sinal, ocorre overflow se o subtrahendo for maior que o minuendo.

O testbench 'testbench.vhd' e o código VHDL 'design.vhd' estão anexados com este relatório. O resultado dos testes por ser conferido na figura 2.