Atividade Formativa 11

Bruno Basseto

6/12/2023

O coração do processador Poli-LegV8 é composto pelos seguintes componentes:

O Banco de Registradores , contendo trinta e dois registradores de 64 bits independentes (**X0** até **X31**), dos quais quaisquer dois registradores podem ser lidos e um terceiro registrador pode ser escrito simultaneamente. O registrador **X31** (também chamado de XZR) é especial e possui sempre o valor da constante *zero*. Os registradores que devem ser lidos são selecionados pelos sinais **Rm** e **Rn**, de cinco bits (de "00000" até "11111") e o registrador a ser escrito é selecionado pelo sinal **Rd** (também de cinco bits). O registrador selecionado por **Rd** somente é escrito na *borda de subida* do sinal de clock, quando o sinal **we** (*write enable*) possuir o valor "1";

A Unidade Lógico-Aritmética (ULA), responsável por executar *operações* de 64 bits com os valores de suas duas entradas, produzindo uma saída com o resultado da operação (64 bits) e uma saída (um bit) indicando se o resultado é nulo (necessário para implementar a instrução CBZ). A operação a ser realizada pela ULA é definida por uma entrada de quatro bits ("*op*"), conforme a tabela 1.

op	Operação
0000	$y = x1\ and\ x2$
0001	$y = x1\ or\ x2$
0010	$y = x1 + x2$
0110	$y = x1 - x2$
0011	$y = x2$
1100	$y = x1\ nor\ x2$

Tabela 1: Operações realizadas pela ULA

As operações "and", "or" e "nor" são realizadas entre cada par de bits das entradas (por exemplo, $y_0 = x_{1_0}\ and\ x_{2_0}$, $y_1 = x_{1_1}\ and\ x_{2_1}$, etc.).

Nas figuras 1 e 2 são apresentados os modelos comportamentais em VHDL para esses dois componentes.

Atividade

1. Implemente uma entidade de nível mais alto (fluxo de dados) em VHDL, interligando o Banco de Registradores e a ULA, para executar as instruções do tipo R (instruções aritméticas e lógicas), como sugerido pela figura 3, a partir da entidade mostrada na figura 4.

2. Crie um *testbench* para gerar os sinais "Rd", "Rn" e "Rm" (que seriam provenientes da instrução atual), "we" e "clk" (que seriam provenientes da Unidade de Controle) para o Banco de Registradores, bem como o sinal "op" para a ULA (que seria proveniente da Unidade de Controle), de forma a simular a execução de algumas instruções do tipo R. Verifique também o correto funcionamento do sinal "zero" e o valor (fixo) de X31. Escreva valores iniciais nos registradores para poder testar, como foi exemplificado na listagem da figura 1.

3. Envie, através do edisciplinas, o seu código VHDL (arquivo ULARegs.vhd) e uma captura da tela da sua simulação (usando o GTKWave ou EDAPlaygroud, em um arquivo PDF, por exemplo). Faça comentários que achar importantes no PDF. Não esqueça de incluir a identificação dos membros do grupo em todos os arquivos enviados.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

--
-- Banco de 32 Registradores de 64 bits
--
entity banco is
port(
   W: in std_logic_vector(63 downto 0);    -- valor de entrada
   Ra: out std_logic_vector(63 downto 0); -- primeira saida
   Rb: out std_logic_vector(63 downto 0); -- segunda saida
   Rd: in std_logic_vector(4 downto 0);    -- indice do registrador a escrever
   Rm: in std_logic_vector(4 downto 0);    -- indice do registrador 1 (ler)
   Rn: in std_logic_vector(4 downto 0);    -- indice do registrador 2 (ler)
   we: in std_logic;                       -- habilitacao de escrita
   clk: in std_logic                       -- sinal de clock
);
end entity;

architecture behavior of banco is
   type REGS is array(0 to 31) of std_logic_vector(63 downto 0);
   signal r: REGS := (
      -- alguns valores iniciais para teste:
      0 => x"000000000000AAAA",     -- I0
      1 => x"0000000000005555",     -- I1
      2 => x"0000000000003333",     -- I2
      3 => x"0000000000000001",     -- I3, etc
      others => (others => '0')
   );
begin
   -- saidas
   Ra <= (others => '0') when Rn = "11111" else
         r(to_integer(unsigned(Rn)));
   Rb <= (others => '0') when Rm = "11111" else
         r(to_integer(unsigned(Rm)));

   -- entrada
   r(to_integer(unsigned(Rd))) <= W
      when (we = '1') and rising_edge(clk);
end architecture;
```

Figura 1: Código VHDL para o Banco de Registradores

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

--
-- ULA para o LegV8
--
entity ULA is
port(
    y: out std_logic_vector(63 downto 0);      -- saida
    x1, x2: in std_logic_vector(63 downto 0); -- entradas
    op: in std_logic_vector(3 downto 0);       -- operacao a realizar
    zero: out std_logic                        -- indica o resultado zero
);
end entity;

architecture behavior of ULA is
    signal op_and, op_or, op_soma,
    op_sub, op_nor: std_logic_vector(63 downto 0);
begin
    -- Operacoes
    op_and <= x1 and x2;
    op_or <= x1 or x2;
    op_soma <= std_logic_vector(unsigned(x1) + unsigned(x2));
    op_sub <= std_logic_vector(unsigned(x1) - unsigned(x2));
    op_nor <= x1 nor x2;

    -- selecao da saida
    with op select y <=
        op_and when "0000",                -- and
        op_or when "0001",                 -- or
        op_soma when "0010",               -- soma
        op_sub when "0110",                -- subtracao
        x2 when "0011",                    -- passa x2
        op_nor when "1100",                -- nor
        (others => '0') when others;       -- outros casos

    -- Verifica resultado igual a zero
    zero <= nor y;
    -- outra opcao
    -- zero <= '1' when unsigned(y) = 0 else '0';
end architecture;
```

Figura 2: Código VHDL para a Unidade Lógico-aritmética

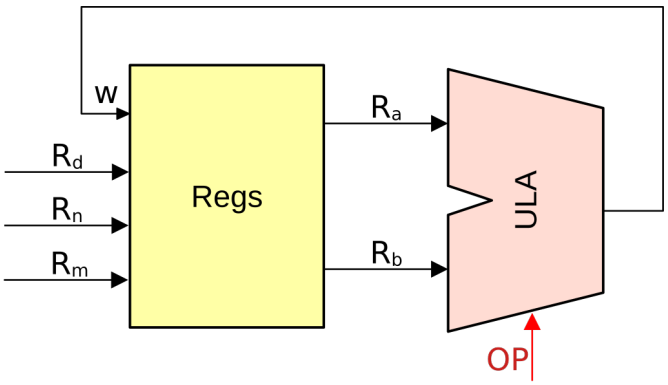

Figura 3: Interligação dos componentes para executar instruções do tipo R

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

--
-- Agrupamento da ULA e Banco de Registradores
--
entity ULARegs is
port(
   y: out std_logic_vector(63 downto 0);      -- saida da ULA
   op: in std_logic_vector(3 downto 0);       -- operacao a realizar
   zero: out std_logic;                       -- indica o resultado zero
   Rd: in std_logic_vector(4 downto 0);       -- indice do registrador a escrever
   Rm: in std_logic_vector(4 downto 0);       -- indice do registrador 1 (ler)
   Rn: in std_logic_vector(4 downto 0);       -- indice do registrador 2 (ler)
   we: in std_logic;                          -- habilitacao de escrita
   clk: in std_logic                          -- sinal de clock
);
end entity;

-- etc...
```

Figura 4: Entidade a ser criada