MCTA024 - SISTEMAS DIGITAIS

Prof. José Artur Quilici-Gonzalez

PROJETO PRÁTICO - ULA

Gustavo Silva de Paula	21085415
Matheus Shimizu Felisberto	21030815
Michele de Camillo Oliveira	11062315

Santo André - SP
2017

1. INTRODUÇÃO

Uma Unidade Lógica e Aritmética (ULA) é uma rede combinacional, que implementa uma função de suas entradas com base em operações lógicas ou aritméticas. Elas constituem o elemento central dos computadores e da maioria dos sistemas digitais.

Nesse projeto foi implementada uma ULA de 3 bits a partir do esquema de uma ULA de 1 bit, sendo feito o código VHDL utilizando sinais correspondentes às saídas das portas lógicas. Foram mantidos os nomes dos sinais das portas lógicas do esquema principal.

2. OBJETIVOS

O objetivo desse projeto é desenvolver uma ULA de 3 bit a partir de três ULA's de 1 bit.

Utilizando o software Quartus II, será criado o código VHDL de uma ULA de 1 bit. A partir do código da ULA de 1 bit, será então feito o código para a ULA de 3 bits. As simulações dos códigos serão realizadas através do ModelSim-Altera. Após as simulações, será feito a adaptação do código VHDL para a placa DE1-Altera e o teste da ULA na mesma.

3. METODOLOGIA

3.1 CÓDIGO VHDL

A partir do diagrama esquemático da ULA de 1 bit, foi feito o código VHDL utilizando diversos *signals* correspondentes a saída de cada porta lógica.

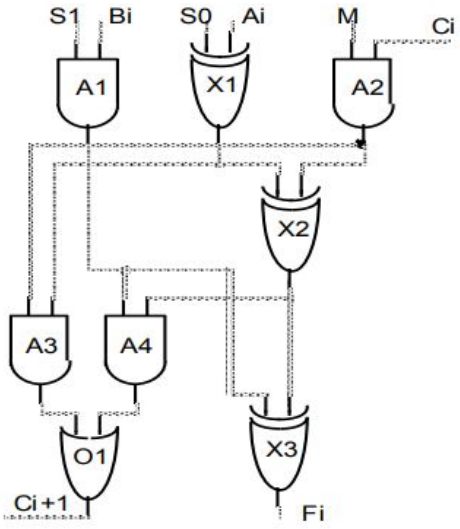

Figura 1 – Diagrama esquemático da ULA de 1 bit. Circuito combinacional

Como as entradas "S0", "S1" e "M" definem qual modo e operação da ULA será utilizado, foi substituído "M" por "S2" para que se utilize apenas um vetor "S" de 3 bits no código VHDL.

A entrada "Ci" foi alterada para "Vem_Um" e a "Ci+1" para "Vai_Um", para um melhor entendimento do funcionamento da ULA.

Os *signals* foram mantidos com o mesmo nome visto em cada porta lógica do circuito. Porém, em vez que criar os *signals* "O1" e "X3", foi atribuído o valor da porta lógica direto nas saídas "Ci+1" e "Fi".

O código VHDL da ULA de 1 bit é uma descrição do fluxo de dados e foi realizado uma simulação funcional utilizando o ModelSim-Altera.

Para construir a ULA de 3 bits foi utilizado 3 ULA's de 1 bit, como mostra o esquema abaixo.

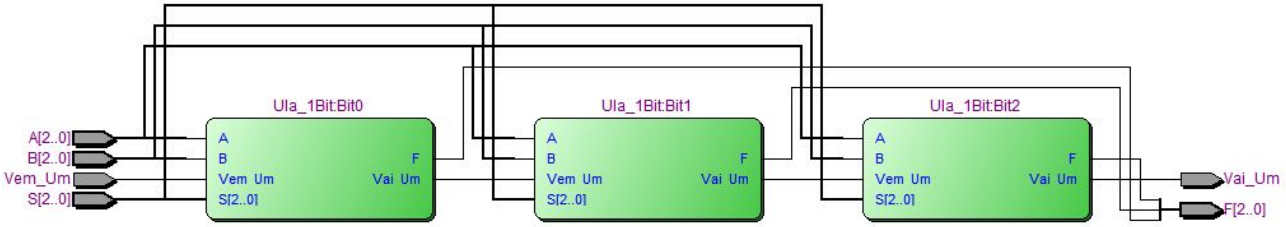

Figura 2 – Esquema da ULA de 3 bits, utilizando 3 ULA's de 1 bit.

Como as operações lógicas e aritméticas são realizadas bit a bit, cada ULA de 1 bit será responsável por fazer a operação de cada bit separado. Pode-se observar no esquema que um bit dos vetores "A" e "B" é enviado para cada ULA. A resposta final será a junção das respostas das 3 ULA's. No caso das operações de soma e subtração, o resultado obtido em uma ULA influência na operação da ULA seguinte, logo, a saída "Vai_Um" de uma ULA é ligada diretamente ao "Vem_Um" da ULA seguinte. O vetor "S", que corresponde ao modo de operação que a ULA realizará, é enviado inteiro para todas as ULA's.

O código VHDL da ULA de 3 bits foi feito criando um componente a partir da ULA de 1 bit e utilizando esse componente para cada um dos 3 bits das entradas "A" e "B". Neste caso, o código VHDL apresenta uma descrição estrutural e a simulação realizada também foi funcional.

Para realizar a simulação utilizando o ModelSim-Altera, foi realizada a síntese lógica, a elaboração e a análise do código VHDL. Também foi criado um código de *test bench* onde foi definido quais os valores das entradas a serem testadas e por quanto tempo a entrada irá ter esse valor.

3.2 PLACA DE1 - ALTERA

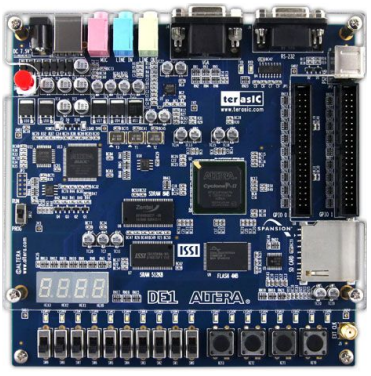

Figura 3 – Placa DE1 - Altera

Para testar a ULA em uma placa DE1 da ALTERA é necessário conhecer o código VDHL. A placa possui como entradas as chaves SW localizadas no canto inferior esquerdo, contando da direita para esquerda a função de cada chave será:

SW(0,1,2) => Entrada S da ULA
SW(5,4,3) => Entrada dos bits B
SW(8,7,6) => Entrada dos Bits A
SW(9) => Entrada *Vem_Um*

Os leds vermelhos logo acima irão representar os valores de entrada para S, B, A. *Vem_Um* respectivamente.

O resultado da operação será representado nos Displays de 7 segmentos, sendo assim da esquerda para direita, cada um irá representar os valores de *A, B, Resultado da Operação.* Apenas para o caso de operações aritméticas, no caso das operações lógicas, o resultado será representado nos Leds Verdes, que ficam do lado direito, sendo os LEDGs 0,1,2 para a o resultado e o LEDG 7 para no caso de estouro na operação, o *Vai_Um*.

Obs: Lembrando que para operação subtração S="111" o Vem_Um deve ser 1 e no caso de um resultado negativo, o Vai_Um será 0.

4. APRESENTAÇÃO DOS DADOS E ANÁLISE DOS RESULTADOS

- ULA de 1 bit:

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity Ula_1Bit is
5   port (
6       Vem_Um, A, B : in STD_LOGIC;
7       S : in STD_LOGIC_VECTOR (2 downto 0);
8       Vai_Um, F : out STD_LOGIC
9       );
10  end Ula_1Bit;
11
12  architecture Comportamental_1 of Ula_1Bit is
13  signal A1, A2, A3, A4, X1, X2 : STD_LOGIC;
14     begin
15        A1 <= S(1) and B;
16        X1 <= S(0) xor A;
17        A2 <= S(2) and Vem_Um;
18        X2 <= X1 xor A2;
19        A3 <= X1 and A2;
20        A4 <= A1 and X2;
21        F <= A1 xor X2;
22        Vai_Um <= A3 or A4;
23  end Comportamental_1;
24
```

Figura 4 – Código VHDL da ULA de 1 bit

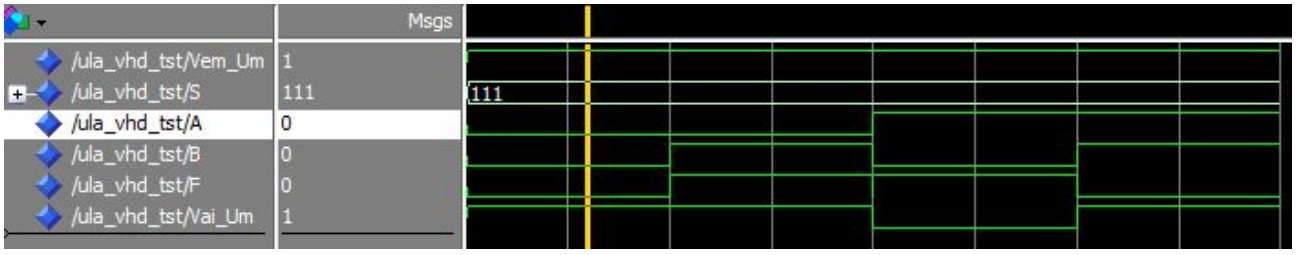

Figura 5 – Simulação funcional da ULA de 1 bit, operação aritmética "B – A" por complemento de 2

- ULA de 3 bits:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Ula_1Bit is
port (
    Vem_Um, A, B : in STD_LOGIC;
    S : in STD_LOGIC_VECTOR (2 downto 0);
    Vai_Um, F : out STD_LOGIC
    );
end Ula_1Bit;

architecture Comportamental_1 of Ula_1Bit is
signal A1, A2, A3, A4, X1, X2 : STD_LOGIC;
   begin
       A1 <= S(1) and B;
       X1 <= S(0) xor A;
       A2 <= S(2) and Vem_Um;
       X2 <= X1 xor A2;
       A3 <= X1 and A2;
       A4 <= A1 and X2;
       F <= A1 xor X2;
       Vai_Um <= A3 or A4;
end Comportamental_1;
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Ula_3Bits is
port (
    Vem_Um : in STD_LOGIC;
    A, B, S : in STD_LOGIC_VECTOR (2 downto 0);
    Vai_Um : out STD_LOGIC;
    F : out STD_LOGIC_VECTOR (2 downto 0)
    );
end Ula_3Bits;

architecture Comportamental_2 of Ula_3Bits is
component Ula_1Bit
    port
    (
       Vem_Um, A, B : in STD_LOGIC;
       S : in STD_LOGIC_VECTOR (2 downto 0);
       Vai_Um, F : out STD_LOGIC
    );
end component;
signal D: std_logic_vector(1 downto 0);
begin
    Bit0: Ula_1Bit
    port map (Vem_Um, A(0), B(0), S, D(0), F(0));
    Bit1: Ula_1Bit
    port map (D(0), A(1), B(1), S, D(1), F(1));
    Bit2: Ula_1Bit
    port map (D(1), A(2), B(2), S, Vai_Um, F(2));
end Comportamental_2;
```

Figura 6 – Código VHDL da ULA de 3 bits. Vale ressaltar que o código da ULA de 1 bit está junto deste código.

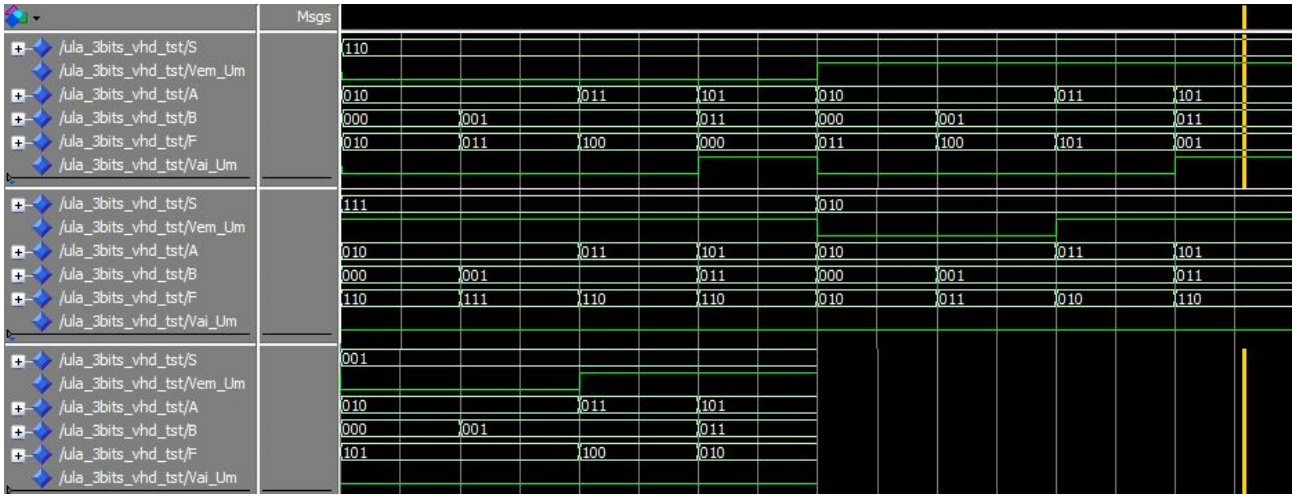

Figura 7 – Simulação funcional da ULA de 1 bit, operações aritméticas: “A+B” (primeira linha) e “B – A” por complemento de 2 (até a metade da segunda linha), operações lógicas: “A xor B” (metade da segunda linha até o final) e “not A” (terceira linha)

Devido a enorme quantidade de possibilidades de valores de entrada, foram simulados apenas alguns casos. Os resultados obtidos pelas simulações correspondem ao esperado, tanto da ULA de 1 bit como da ULA de 3 bits.

5. TUTORIAL DE FUNCIONAMENTO DO PROGRAMA

5.1 SIMULAÇÕES UTILIZANDO O MODELSIM-ALTERA

Com o código pronto, é necessário realizar a síntese lógica, a elaboração e a análise do código. Essas operações podem ser encontradas em: **Processing** => **Start**, na barra de ferramentas.

Após realizar essas operações, é necessário gerar um modelo de *test bench*. No mesmo caminho utilizado anteriormente há as opções "*Start Partition Merge*" e "*Start Test Bench Template Writer".* essas operações devem ser realizadas exatamente nesta ordem. Com isso foi criado o modelo do *test bench* dentro do diretório *"simulation/modelsim"* na pasta raiz do projeto.

Abra o modelo através de: **File** => **Open…**, não esquecendo de selecionar o tipo de arquivo como *"Test Bench Output Files"*.

Com o test bench aberto, há um processo abaixo do "PORT MAP" reservado para inserir os estímulos, ou vetores, de entrada.

Salve o arquivo e rode o ModelSim-Altera indo em: **Tools** => **Run Simulation Tool** => **RTL Simulation.**

Vá em **Compile** => **Compile**, selecione seu arquivo de test bench, aperte em "*compile*" e depois "*done*". Uma *"entity"* com o nome do test bench deve aparecer na biblioteca *"work"* nesta janela. Após dar duplo clique na *"entity"* uma nova janela no simulador irá aparecer. Adicione todos os itens na janela da esquerda para a onda clicando com o botão direito do mouse e indo em: **Add** => **To Wave** = > **All items in region.** Clique em *"Run All"* presente na barra de ferramentas para rodar a simulação e depois em *"Zoom Full",* para ajustar a escala.

5.2 PLACA DE1 - ALTERA

Para realizar o upload do programa para a placa devemos seguir os passos a seguir:

Na parte esquerda do Quartus II, na seção "**Task"**, dê dois cliques na opção **"Program Device (Open Programmer)"**, a janela deve aparecer, verifique se as opções **"USB-Blaster"** e **"Mode: JTAG"** estejam ajustadas, então clique em **"Add File"**, escolha o caminho **"output files"** e selecione o arquivo **".sof"**. Com a placa DE1 Altera conectada e **ligada** clique em **"Start"**, e aguarde aparecer **"Progress: 100% (Succesful)"** .

6. CONCLUSÃO

Após a realização deste projeto, definimos que a criação de uma ULA de 3 bits por meio da combinação de três ULAs de 1 bit pode ser concluída por meio de uma série de procedimentos como programação de código em VHDL, simulações, configurações de valores de entrada.

Foi necessário primeiramente desenvolver o código VHDL para uma ULA de 1 bit e então fazer uma simulação utilizando o ModelSim-Altera.

Partindo da simulação, foi desenvolvida uma ULA de 3 bits e simulada também utilizando o ModelSim-Altera.Como última etapa, a ULA de 3 bits foi implementada na placa DE1 da ALTERA. Os resultados obtidos para a implementação e simulação da ULA de 1 bit e da ULA de 3 bits foram satisfatórios.