Projeto de Unidade Lógica Aritmética

O presente trabalho tem por objetivo explorar conceitos de circuitos combinacionais explorados em sala de aula. Para tanto, deverão ser desenvolvidos módulos VHDL mesclando descrições estruturais e comportamentais, bem como deve ser implementado um módulo para validação do circuito (*testbench*).

Descrição do módulo de hardware a ser implementado

A Figura 1 ilustra o módulo a ser implementado, que é composto por uma unidade lógica aritmética (ULA) e dois codificadores (COD) para a conversão de 4 bits para uma representação em um display de 14 segmentos. A ULA tem como entradas dois operandos (A e B) de 8 bits e um seletor da operação a ser realizada (Op) de 3 bits; e tem como saídas o sinal S de 8 bits, representado o resultado da operação e os qualificadores de Zero (Z), Negativo (N), Carry (C) e Overflow (O). A saída da ULA é bipartida de forma a gerar duas entradas de 4 bits (SH e SL) para os COD s. Cada COD, por sua vez, produz uma saída para alimentar um display de 14 segmentos. Sendo H a saída do display que representa o algarismo mais significativo e L a saída do display que representa o algarismo menos significativo.

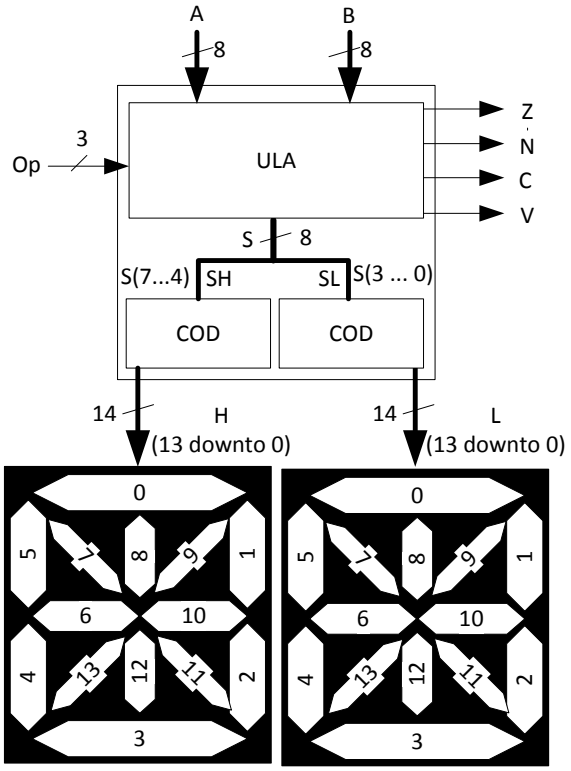

Figura 1 - Diagrama de blocos do da solução a ser implementada.

A ULA deverá permitir a escolha de uma das operações listadas na Tabela 1 conforme codificação de três bits duas últimas colunas da tabela. A construção da ULA deverá ser feita de forma estrutural, sendo então uma construção hierárquica, onde os módulos "folha" deverão representar a operação bit a bit. Como exemplo, o módulo que deverá ser construído para implementar a soma, terá como elemento básico de implementação somadores de 1 bit. Para a geração dos qualificadores deverá ser feita de forma comportamental.

Tabela 1 - Operações suportadas pela ULA.

Operação	Tipo da instrução	Codificação	
		bit 2	bits 1 e 0
Soma	**Aritméticas**	**0**	**00**
Subtração			**01**
Complemento de 2			**10**
Paridade			**11**
Complemento	**Lógicas**	**1**	**00**
Não-Ou lógico			**01**
Ou exclusivo lógico			**10**
Não-E lógico			**11**

Para a construção do somador, deverá ser implementado um somador de propagação rápida, cujo diagrama de blocos é apresentado na figura que segue. Na figura, é apresentado um somador de 4 bits. O somador solicitado no presente trabalho é um é de 8 bits. Este deve ser implementado a partir da instanciação de 2 somadores de propagação rápida de 4 bits, equivalente ao apresentado. Neste diagrama, o carry out (c4) e o resultado da soma (S0...S3) são portas de saída, enquanto o carry in (c0) e os operando (B0...B3; A0...A3) são portas de entrada.

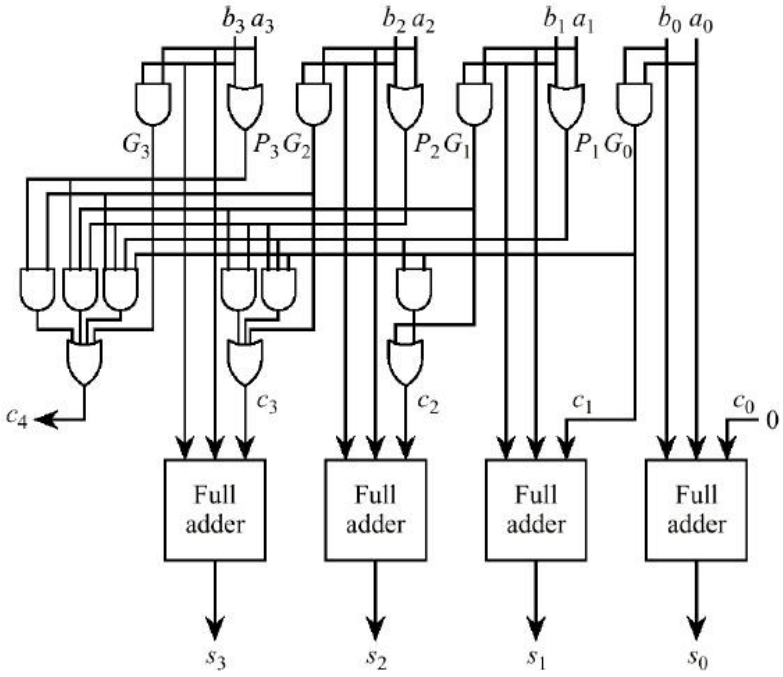

Figura 2 - Somador de propagação rápida.

Para a implementação do **subtrator**, deve-se utilizar o somador com as entradas sendo adaptadas. A operação a ser realizada é A – B. Para isto, o operando B deve ter seu sinal invertido, o que é alcançado com complemento de 2. Para a implementação da operação de **complemento de 2**, o somador também deve ser usada. A operação de **complemento de 2** e cálculo de **paridade** devem ser realizadas sobre o operando A. O cálculo de paridade deve retornar "0000 0001" caso o número de bits em '1' na entrada A seja ímpar e "0000 0000" se o número de bits em '1' seja 0.

Para as operações lógicas, a operação de complemento, ou seja a de inversão de todos os bits, deve ser realizada sobre o operando A. Todas as demais operações lógicas serão realizadas utilizando-se os dois operandos.

Para a construção do codificador do display de 14 segmentos, considere que os segmentos são acesos quando o valor lógico é ‘1’ e desligados quando o valor lógico é ‘0’. Lembre que com 4 bits podemos representar até 16 valores, que em hexadecimal variam de 0 a F. A Figura 2 ilustra como os valores hexadecimais deverão ser representados no display. A construção do codificador pode ser feita de forma comportamental.

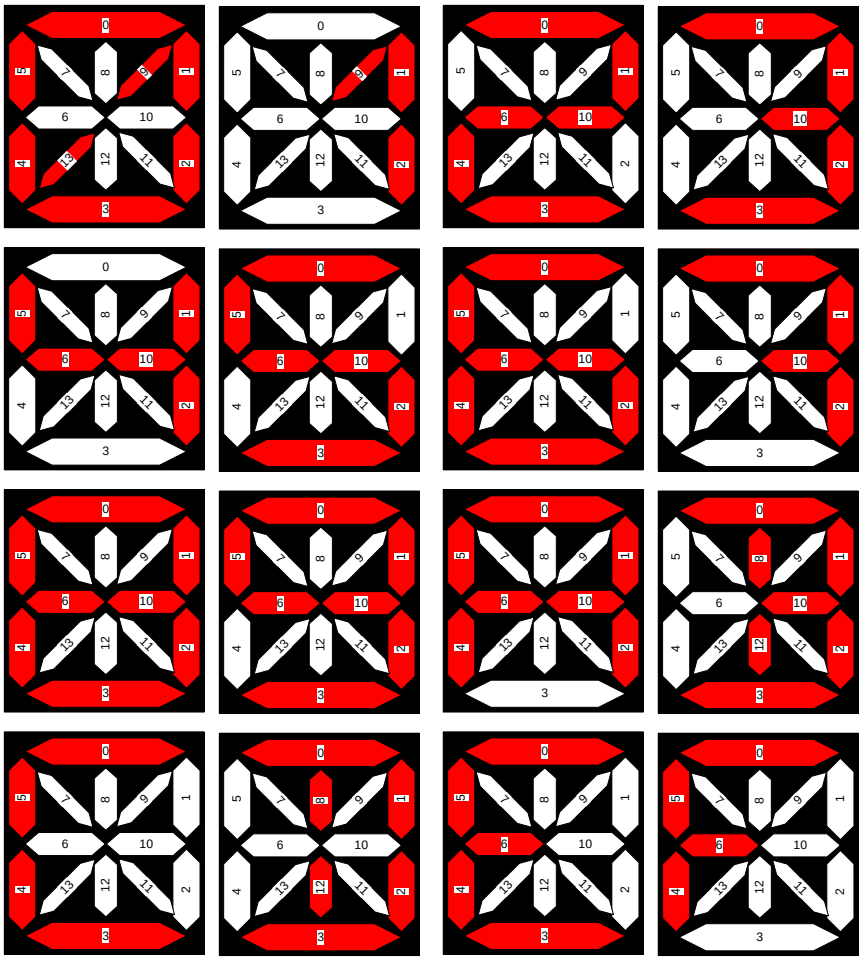

Figura 3 - Codificação a ser empregada para o display de 14 segmentos.

A seguinte entidade é esperada do módulo ilustrado na Figura 1.

```
entity meuModulo is
	port
	(
		A, B: in std_logic_vector(7 downto 0);
		Op: in std_logic_vector(2 downto 0);
		H, L: out std_logic_vector(13 downto 0);
		N, Z, C, V: out std_logic
	);
end meuModulo;
```

Validação e documentação

Para a validação do módulo que deverá ser implementado deve-se gerar um *testbench*. Este deverá permitir validar todas as operações (lógicas e aritméticas), bem como a variação dos qualificadores e a codificação do display de segmentos. A escolha dos padrões de estímulos a serem gerados deverá ser documentada em relatório, deixando claro o que se espera como resultado daquele estímulo. Adicionalmente, algumas imagens da simulação deverão ser apresentadas em relatório com o objetivo de comprovar se o que se esperava como resultado foi efetivamente alcançado. Para cada imagem postada no relatório, espera-se a inserção de comentários.

Além destas informações, um diagrama de blocos deverá estar presente no documento, deixando claro que sub-módulos compõem cada módulo.

Considerações finais

O trabalho pode ser realizado em grupos de até três (3) alunos. Este deve ser entregue até o dia determinado na agenda da disciplina, via moodle, antes do início da aula.

A entrega deve ser feita com um arquivo compactado (.zip) onde deverão estar contidos os arquivos VHDL do projeto e o relatório (em PDF). O nome do arquivo deve conter nome e sobrenome de todos os alunos que fizeram parte do trabalho. Ex.: AnaMattos_CarlosBrandao_JanainaMirten.zip.

O material postado no moodle é de inteira responsabilidade do aluno. A presença de arquivos corrompidos, que impeçam a avaliação do trabalho pelo professor será considerada como a não entrega do trabalho.