UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA - CT
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CIRCUITOS DIGITAIS

UNIDADE LÓGICA ARITMÉTICA (ULA)

ELE2715 - Grupo 03 - Projeto - Problema 02

Albertho Siziney Costa
Eduardo Garcia Zaccharias
Igor Michael Araujo de Macedo
Isaac de Lyra Junior
João Matheus Bernardo Resende

Natal, 04 de fevereiro de 2021

Albertho Siziney Costa
Eduardo Garcia Zaccharias
Igor Michael Araujo de Macedo
Isaac de Lyra Junior
João Matheus Bernardo Resende

UNIDADE LÓGICA ARITMÉTICA (ULA)

Relatório técnico apresentado como requisito parcial para obtenção de aprovação na disciplina ELE2715 - Circuitos Digitais, do Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Norte.

Docente: Prof. Dr. Samaherni Morais Dias

Natal, 04 de fevereiro de 2021

RESUMO

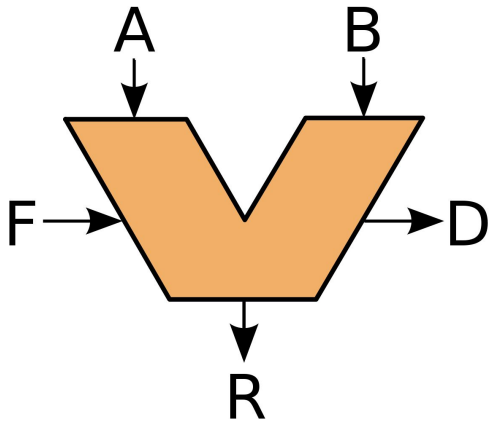
Este relatório tem como objetivo desenvolver um projeto de um circuito lógico combinacional que implemente uma Unidade Lógica Aritmética (ULA). Este terá como entrada dois números de 8 bits, irá efetuar uma operação lógica aritmética entre eles e, em seguida, realizar uma codificação binária, conhecida como BCD, para exibir o resultado, em decimal, em três *displays* de 7 segmentos. Aqui, portanto, será analisado como executar cada operação, fazer a codificação e apresentação do resultado em *display*. O projeto contemplará uma solução utilizando circuitos integrados (CIs) comerciais.
Palavras-chave: Circuitos digitais. Tabela-verdade. ULA. BCD. *Display* 7 Segmentos.

SUMÁRIO

1. INTRODUÇÃO

Uma Unidade Lógica e Aritmética (ULA) é um componente fundamental da unidade central do processador e até dos mais simples microprocessadores. Ela combina uma variedade de operações lógicas e matemáticas, dentro de uma única unidade, por meio de um circuito combinatório. As operações são feitas entre duas entradas de dados com *N* bits de largura e gerando uma saída de dados de *N* bits. Dentre as operações, estão: adição, subtração e operações lógicas como *AND, NOT, OR* e *XOR*. Na Figura 1 é possível ver como os processos geralmente acontecem, podendo haver variações. As entradas *A* e *B* são os operandos, *F* é a entrada de controle, que determina qual operação será realizada, *R* é a saída resultante da operação e *D* pode ser utilizada para diversas utilidades.

Figura 1: Esquemático para uma ULA.



Fonte: Wikipédia.

Após realizar a operação desejada com a ULA, o circuito proposto tem como objetivo exibir o resultado em *displays* de 7 segmentos, sendo necessário, assim, converter o número de binário para decimal. Para isso, será utilizado um método conhecido como codificação binária decimal - BCD (*binary coded decimal*), no qual cada dígito do número em decimal é representado por seu equivalente em binário (TOCCI, 2011). No caso da ULA em questão, serão realizadas operações entre entradas de 8 bits, resultando em uma saída também de 8 bits, que, em decimal, podem ser números de até 3 dígitos, sendo assim, após a codificação, o valor em BCD terá 12 bits. Na figura 2 é possível ver dois exemplos.

Figura 2: Exemplos de conversão BCD.

8	7	4	(decimal)	9	4	3	(decimal)
↓	↓	↓		↓	↓	↓	
1000	0111	0100	(BCD)	1001	0100	0011	(BCD)

Fonte: (TOCCI, 2011).

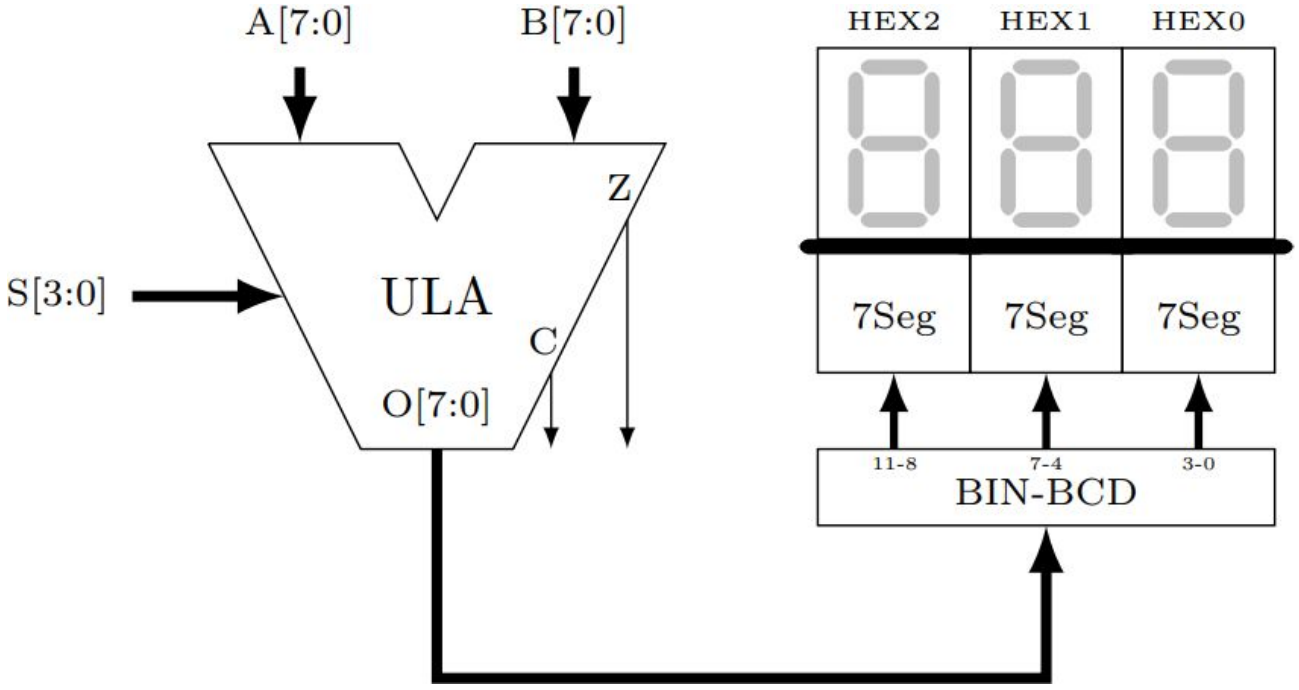
Por fim, o passo final será mostrar cada um dos dígitos do número, em decimal, em um *display* de 7 segmentos, que recebe como entrada um número em binário de 4 bits e retorna o número correspondente em decimal.

2. DESENVOLVIMENTO

O esquemático do circuito pode ser visto na Figura 3. Para a ULA, haverá duas entradas, *A* e *B*, de 8 bits, que serão os dados a serem operados, uma entrada de controle, *S*, de 4 bits, que indicará qual operação/função será selecionada, uma saída, *O*, de 8 bits, que será o resultado da operação e, por fim, duas saídas, *C* e *Z*, de 1 bit, que representarão o *Carry* e o *Zero*, respectivamente. A relação de operações pode ser vista na Figura 4.

Por fim, a saída *O*, da ULA, será levada ao codificador *BIN-BCD*, que, ao convertê-la, será levada aos *displays* de 7 segmentos para, assim, representar o resultado da operação em decimal.

Figura 3: Esquemático do circuito.



Fonte: Dados do problema.

Figura 4: Conjunto de instruções do circuito da ULA.

Função S[3:0]	**Instrução**	**Descrição**	**Carry**	**Zero**
1	ADD A, B	O=A+B	•	•
2	SUB A, B	O=A-B	•	•
3	INC A	O=A+1	•	•
4	DEC A	O=A-1	•	•
5	MUL A, B	O=(A·B)[7:0]	•	•
6	AND A, B	O=A AND B		•
7	OR A, B	O=A OR B		•
8	XOR A, B	O=A XOR B		•
9	NOT A	O=not(A)		•
10	SHL A, B	O=A≪B[2:0], C=Msb	•	•
11	SHR A, B	O=A≫B[2:0]		•

Fonte: Dados do problema.

2.1 Somador

A operação soma de binários é possível ser feita utilizando dois componentes combinacionais, sendo um chamado meio somador e outro chamado somador completo (VAHID, 2008). Eles serão utilizados em sequência, dependendo do número de bits que terá a adição principal. O meio somador (MS) é composto por duas entradas de 1 bit cada e duas saídas também de 1 bit cada. As entradas, *a* e *b*, representam os bits a serem somados, a saída, *s*, o resultado da adição e a última saída, *co*, o chamado *carry out*, que funciona como o "vai um", técnica utilizada, normalmente, para cálculos de adição feitos "a mão". No Tabela 1 é possível ver a Tabela verdade para um meio somador.

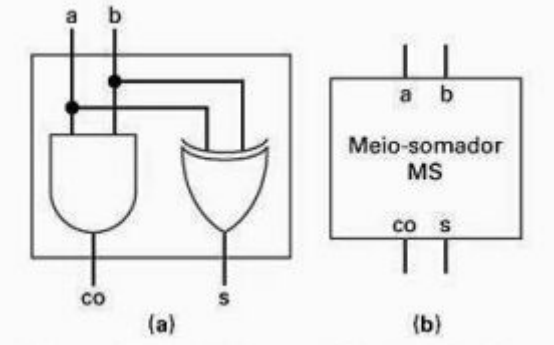
Tabela 1: Tabela verdade de um meio somador.

Entradas		**Saídas**	
a	b	co	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Fonte: Autores.

Ao analisar a tabela, é possível perceber que a saída *s* tem nível alto quando as entradas são diferentes, característica de uma porta *OR* exclusiva (*XOR*), já a saída *co* tem o comportamento de uma porta *AND*. Sendo assim, um meio somador pode ser representado assim como na Figura 5, onde, na esquerda, tem-se a representação do circuito e, na direita, o símbolo para diagrama de blocos.

Figura 5: Meio somador: (a) circuito e (b) símbolo para diagrama de blocos.



Fonte: (VAHID, 2008).

Já para o somador completo (SC), há três entradas e duas saídas. Ele soma três bits, *a, b* e *ci*, sendo o último, *ci*, o *carry in*, resultado do "vai um" da operação anterior, gera uma soma, *s*, e um bit de transporte "vai um"/ *carry out*, *co*. A tabela verdade para este tipo de somador pode ser vista na Tabela 2.

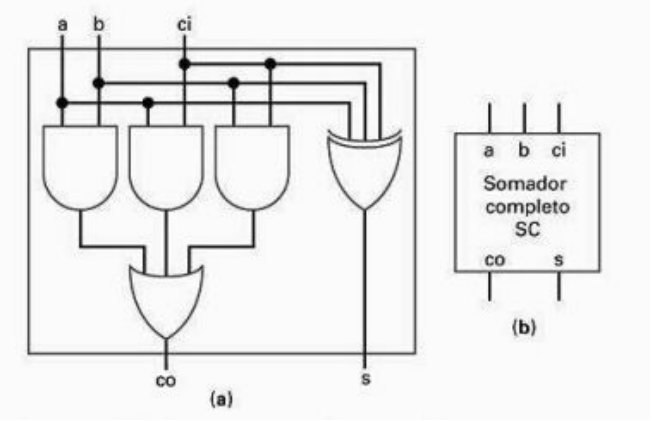
Tabela 2: Tabela verdade de um somador completo.

Entradas			Saída	
a	b	ci	co	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Fonte: Autores.

Tendo a tabela-verdade em mãos, é possível retirar o circuito equivalente para ela. Ele pode ser visto na Figura 6.

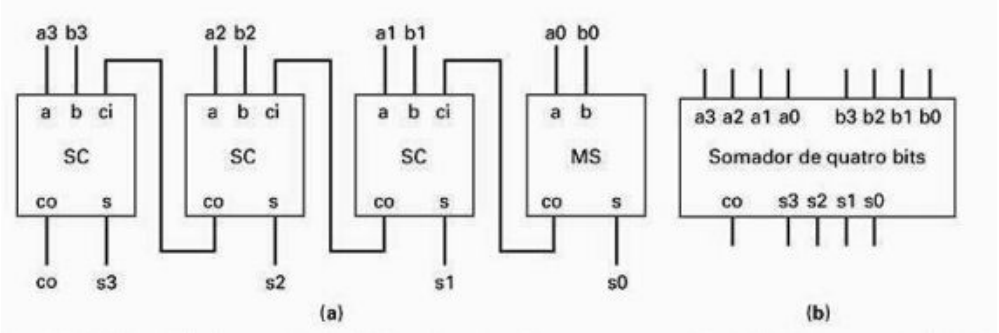
Figura 6 : Somador completo: (a) circuito e (b) símbolo para diagrama de blocos..



Fonte: (VAHID, 2008).

Sendo assim, para implementar um somador basta unir um meio somador seguido de vários somadores completos, dependendo do número de bits que deseja-se somar. Na Figura 7, por exemplo, é possível ver como seria a implementação para um somador de 4 bits.

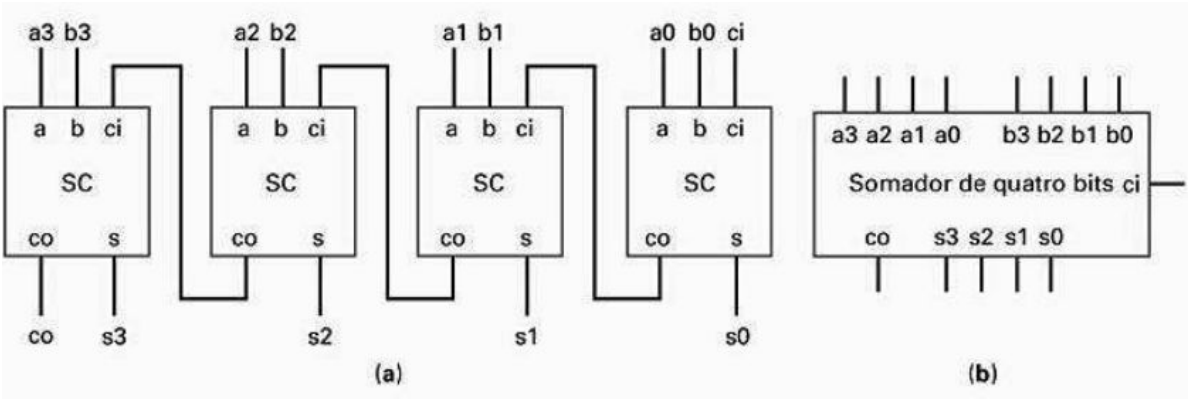
Figura 7 : Somador de quatro bits: (a) implementação usando três somadores completos e um meio somador e (b) símbolo para diagrama de blocos.



Fonte: (VAHID, 2008).

Para a implementação de um somador de 8 bits, uma solução seria utilizar 1 meio somador e 7 somadores completos. Outra solução seria ainda, usar somente 4 SC, para somar 4 bits, sendo o primeiro *ci = 0*, e unir esse somador com outro de 4 bits, onde o primeiro *ci* seria o *co* do último SC do bloco anterior. A implementação para esse tipo de somador de 4 bits pode ser vista na Figura 8.

Figura 8: Somador de quatro bits: (a) implementação com 4 somadores completos e (b) símbolo para diagrama de blocos.



Fonte: (VAHID, 2008).

Por fim, o somador será acionado quando na ULA for inserida a entrada de controle $S_{ADD} = 0001$.

2.2. Subtrator

Podemos entender o subtrator, assim como no somador, como módulos de operação, dessa vez, utilizando meio subtrator e subtrator completo. O meio subtrator pode ser entendido como uma operação entre dois bits sem receber um empréstimo anterior, por esse motivo ele é a melhor escolha para operação de 2 bits. A tabela a seguir mostra a tabela verdade para um meio subtrator:

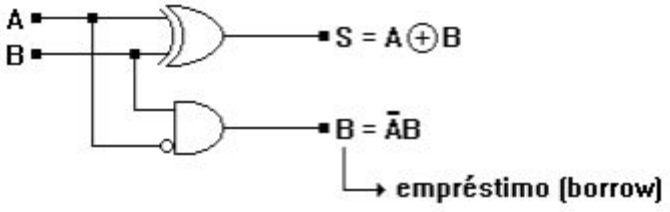
Tabela 3 : Tabela verdade para um meio subtrator.

Entradas		**Saídas**	
a	b	b	s
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Fonte: Autores;

Analisando a tabela acima, é possível perceber que a saída *s* tem nível alto para entradas diferentes, esse comportamento é de uma porta *exclusive or* (*xor*), e a saída *b* de empréstimo tem nível alto quando (*not* A) *and* B. Dessa forma, o circuito lógico correspondente pode ser visto na Figura 9.

Figura 9 : Circuito lógico meio subtrator.



Fonte: Autores.

Em contrapartida, o subtrator completo leva em consideração o empréstimo anterior. Sua tabela verdade pode ser vista a seguir:

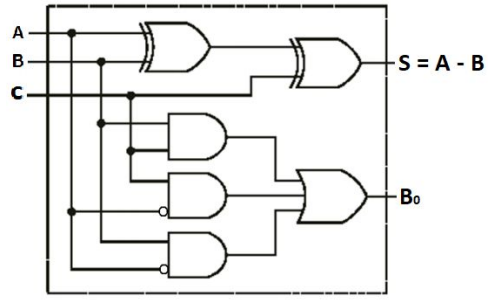
Tabela 4 : Tabela verdade subtrator completo.

Entradas			Saída	
a	b	ci	B	s
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Fonte: Autores.

Analisando a tabela verdade, é possível retirar o circuito lógico e que pode ser visto na Figura 10.

Figura 10 : Circuito lógico para um subtrator completo.



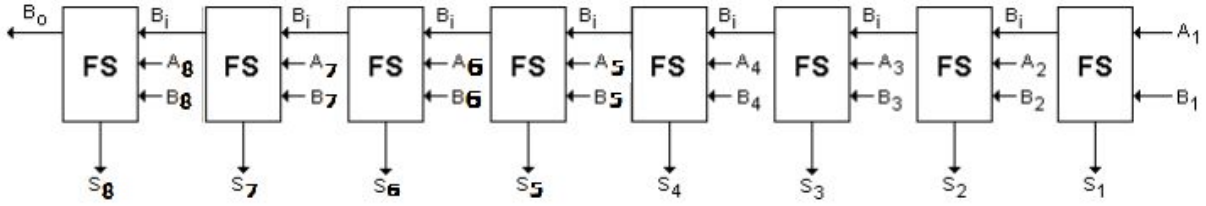
$S = A \oplus B \oplus C \qquad B = \overline{A}C + \overline{A}B + BC$

Fonte: Autores.

No circuito acima pode-se observar agora três entradas, onde A e B são os bits a serem operados e a variável C é bit do empréstimo anterior.

De acordo com a situação-problema proposta na atividade, uma possível solução para a realização da subtração bit a bit, de um número de oito bits, será a implementação de oito blocos de subtratores completos ligados em paralelo, como mostra a figura abaixo:

Figura 11 : Subtrator de 8 bits.



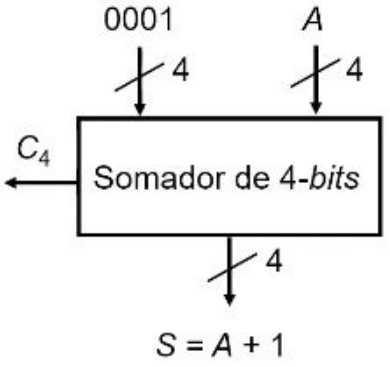
Fonte: Autores.

Por fim, o subtrator será acionado quando na ULA for inserida a entrada de controle $S_{SUB} = 0010$.

2.3 Incrementador e Decrementador

O incrementador, como o próprio nome sugere, é um componente combinacional que adiciona + 1 no dado de entrada, ou seja, insere o valor binário 1 para ser somado no final do vetor de bits de entrada. O incrementador/decrementador é utilizado em contadores para alternar os valores do vetor de bits. A princípio, o incrementador é uma aplicação específica do circuito somador. Pegando como exemplo um circuito somador de 4-bits, seria possível criar um incrementador apenas forçando uma das entradas como sendo 0001, na Figura 12 é possível visualizar a implementação de um incrementador utilizando somador de 4-bits.

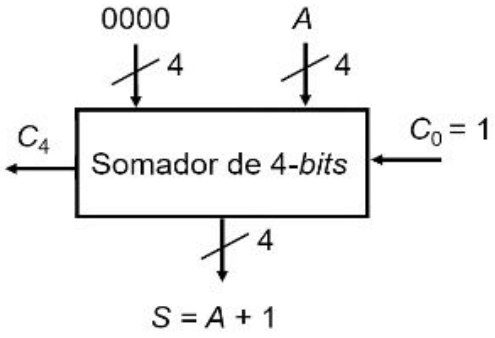
Figura 12: Incrementador com somador de 4-bits.



Fonte: Autores.

Dentro do contexto da aplicação de somadores de 4-bits funcionando como incrementadores, podemos mencionar também outra configuração, sendo o *carry* de entrada $C_0 = 1$ e uma das entradas sendo 0000, tal configuração é demonstrada na Figura 13.

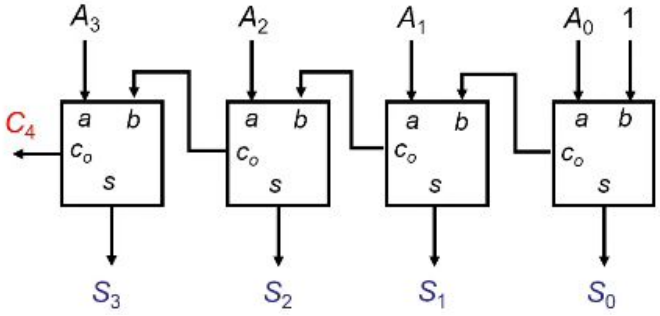
Figura 13: Incrementador utilizando somador de 4-bits e $C_0 = 1$



Fonte: Autores.

Por fim, a maneira mais compacta de construir um incrementador é utilizando apenas meio-somadores em cascata, para um incrementador de 4-bits é necessário utilizar 4 meio-somadores onde nas entradas A_i são colocados os bits do dado que queremos incrementar e no meio-somador menos significativo é colocado $b = 1$. Na Figura 14 é exibido um esquemático dessa configuração.

Figura 14: Meio somadores em cascata na configuração de incrementador.

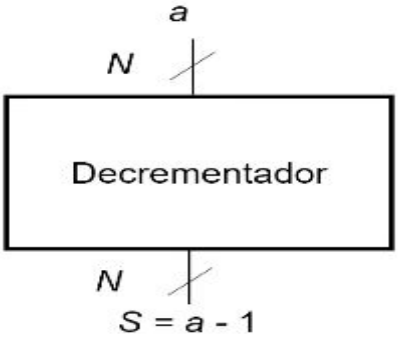


Fonte: Autores.

Com tudo em mente para resolução de um incrementador, basta escolher uma das configurações acima, onde é utilizado o circuito somador de uma forma específica. Por fim, o incrementador será ativado quando a entrada de controle da ULA for $S_{INC} = 0011$.

Para o decrementador, o circuito irá receber um dado de entrada A e sua saída será a subtração binária $S = A - 1$. A lógica para implementação do decrementador será a mesma do incrementador, exceto pelo fato que serão utilizados subtratores ao invés de somadores. Sendo assim, é possível utilizar as 3 configurações mencionadas anteriormente.

Figura 15: Esquemático de um decrementador.



Fonte: Autores.

Por fim, o decrementador será acionado quando na ULA for inserida a entrada de controle $S_{DEC} = 0100$.

2.4 Multiplicador

O circuito multiplicador tem o mesmo princípio da multiplicação decimal, porém, há alguns pontos importantes a serem destacados, observe a figura abaixo:

Figura 15 : Princípio da multiplicação.

				a3	a2	a1	a0
			x	b3	b2	b1	b0
				a3b0	*a2b0*	*a1b0*	*a0b0*
			a3b1	*a2b1*	*a1b1*	*a0b1*	
		a3b2	*a2b2*	*a1b2*	*a0b2*		
+	*a3b3*	*a2b3*	*a1b3*	*a0b3*			
p7	p6	p5	p4	p3	p2	p1	p0

Fonte: Autores.

É possível perceber que a saída *p0* pode ser obtida utilizando uma porta *AND* entre *a0* e *b0*. Já para *p1*, basta utilizar um somador completo entre os valores de *a1b0* e *a0b1*. Vale lembrar que essa soma pode gerar um *carry out*, dessa maneira, esse *carry* será somado juntamente com *a2b0*, *a1b1* e *a0b2*. Podemos observar que teremos uma soma em cascata implementada com somadores e portas *AND*, até completar os oito bits de saída. A última saída *p7* poderá ser o *carry out* proveniente da soma de *p6,* caso essa receba um *carry* e a combinação de *a3b3* for igual a 1.

Por fim, o multiplicador será acionado quando na ULA for inserida a entrada de controle $S_{MUL} = 0101$.

2.5 *AND*, *OR*, *XOR* e *NOT*

Para as operações lógicas *AND*, *OR*, *XOR* e *NOT*, elas serão ativadas quando a entrada de controle for: $S_{AND} = 0110$, $S_{OR} = 0111$, $S_{XOR} = 1000$ e $S_{NOT} = 1001$. Elas funcionam aplicado a operação lógica correspondente a cada uma dos 8 bits das entradas *A* e *B*. Na Tabela 5 é possível ver como ocorre.

Tabela 5: Operações lógicas *AND, OR*, *XOR* e *NOT*.

Entrada		***Saída***			
A	*B*	SAND=0110	*SOR=0111*	*SXOR=1000*	*SNOT=1001*
A(0)	B(0)	O(0) = A(0) and B(0)	O(0) = A(0) or B(0)	O(0) = A(0) xor B(0)	O(0) = *not* A
A(1)	B(1)	O(1) = A(1) and B(1)	O(1) = A(1) or B(1)	O(1) = A(1) xor B(1)	O(1) = *not* A
A(2)	B(2)	O(2) = A(2) and B(2)	O(2) = A(2) or B(2)	O(2) = A(2) xor B(2)	O(2) = *not* A
A(3)	B(3)	O(3) = A(3) and B(3)	O(3) = A(3) or B(3)	O(3) = A(3) xor B(3)	O(3) = *not* A
A(4)	B(4)	O(4) = A(4) and B(4)	O(4) = A(4) or B(4)	O(4) = A(4) xor B(4)	O(4) = *not* A
A(5)	B(5)	O(5) = A(5) and B(5)	O(5) = A(5) or B(5)	O(5) = A(5) xor B(5)	O(5) = *not* A
A(6)	B(6)	O(6) = A(6) and B(6)	O(6) = A(6) or B(6)	O(6) = A(6) xor B(6)	O(6) = *not* A
A(7)	B(7)	O(7) = A(7) and B(7)	O(7) = A(7) or B(7)	O(7) = A(7) xor B(7)	O(7) = *not* A

Fonte: Autores.

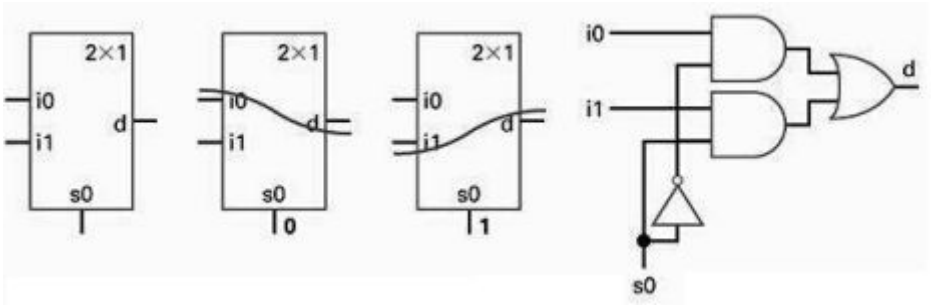
2.6 Deslocador à esquerda

A ideia proposta para a implementação do deslocador à esquerda é estabelecer uma saída de 8 bits resultante de um possível deslocamento dos mesmos, onde o incremento desse deslocamento será definido através dos 3 bits mais significativos da entrada *B* da ULA. Nesse caso, como são 3 bits, é possível haver um deslocamento de 0 bits (*B=000*) até de 7 bits (*B=111*).

Seguindo a ideia vista no livro Vahid (2008, p.191), é possível criar um deslocador utilizando dispositivos multiplexadores 2x1. Esses dispositivos possuem uma saída, duas entradas

de dados e uma de seleção. Sua lógica de funcionamento é “setar” para a saída, uma das entradas de dados, de acordo com a entrada de seleção.

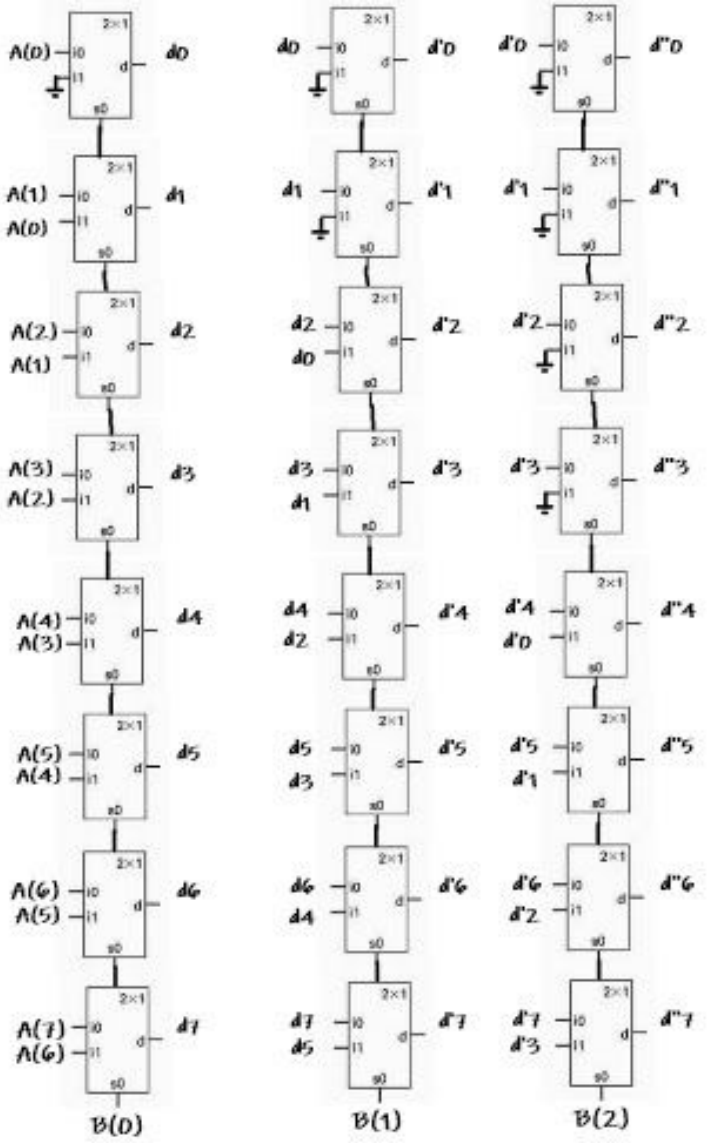
Figura 16: Multiplexador (2x1).



Fonte: (VAHID, 2008).

Como pode ser visto na Figura 21, a saída *d* recebe uma das entradas *i0* e *i1*, de acordo com a entrada selecionada em *s0*. Sendo assim, é possível implementar o funcionamento do deslocador através de uma cascata de multiplexadores que irão realocar os valores “1” ou “0” em um determinado espaço da entrada *A* de acordo com a quantidade de deslocamentos definida por *B*

Figura 17 : Deslocamento à esquerda utilizando multiplexadores 2x1..



Fonte: Autores.

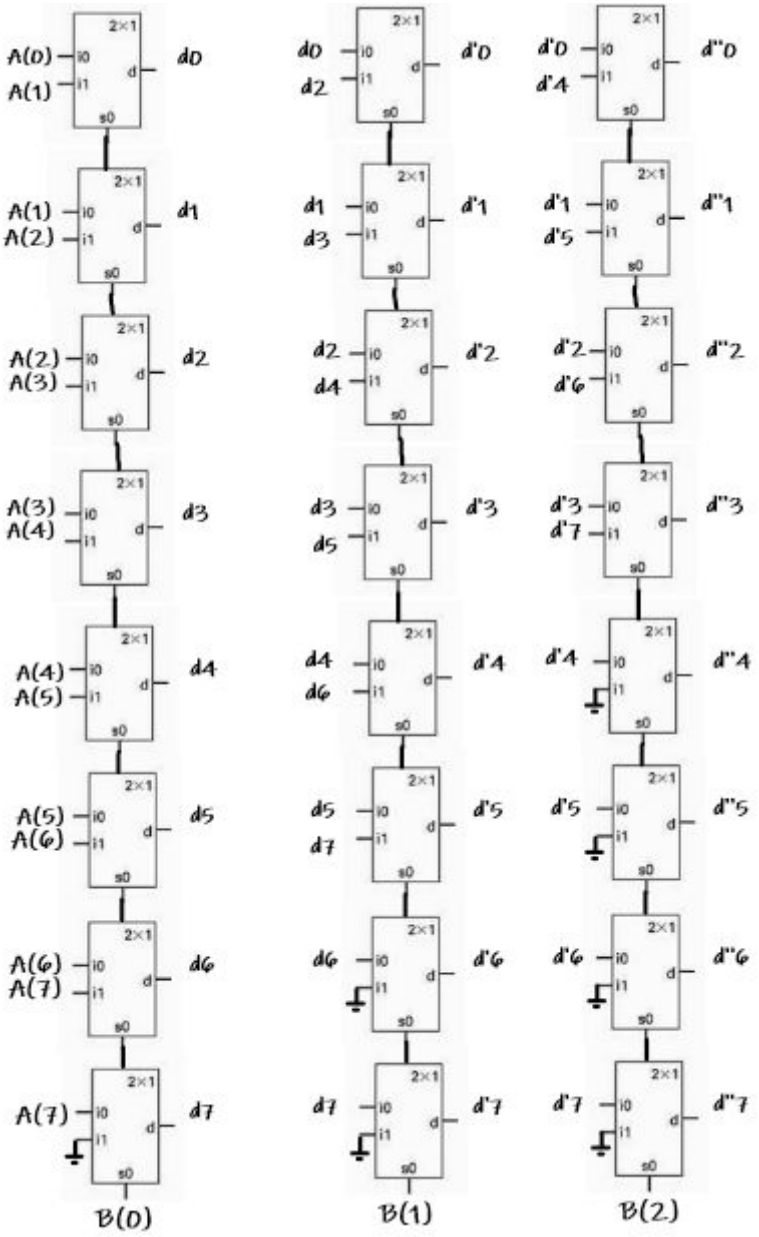
A Figura 22 mostra que *B* vai selecionar as respectivas colunas de acordo com seu valor lógico. Por exemplo, se *B(0)* for 0, a primeira coluna irá atuar com o *s0* recebendo 0. Note que na primeira coluna há um aterramento indicando o nível lógico 0 em *i1* no primeiro multiplexador. Isso ocorre porque ao deslocar o número binário para a esquerda, o valor menos significativo irá receber 0, e os valores seguintes receberão os valores anteriores. Perceba também que esse deslocamento ocorre para *i1* = 1, pois quando não houver deslocamento de um determinado bit, o *d0* receberá o próprio *A(0)* e assim por diante. Na segunda coluna, existem 2 aterramentos, pois se *B(1) = 1*, haverá dois incrementos de deslocamento, zerando os dois bits menos significativos. A mesma lógica serve para a terceira coluna. Após essa implementação, as saídas *d''* serão o vetor binário deslocado à esquerda.

Por fim, o deslocador à esquerda será acionado quando na ULA for inserida a entrada de controle $S_{SHL} = 1010$.

2.7 Deslocador à direita

A lógica para o deslocador à direita pode seguir a mesma ideia do deslocador à esquerda, utilizando multiplexadores. Mas, nesse caso, os aterramentos serão conectados nos últimos segmentos da entrada, como é mostrado na Figura 18.

Figura 18: Deslocador à direita utilizando multiplexadores.



Fonte: Autores.

Por fim, o deslocador à direita será acionado quando na ULA for inserida a entrada de controle $S_{SHR} = 1011$.

2.8 Representação dos resultados

O último passo, após a realização da operação escolhida na ULA, é a representação do resultado. Voltando a observar o esquemático do circuito, na Figura 3, para entender o fluxo, observa-se que a saída *O* é levada até a entrada do *BIN BCD*. Isto porque o objetivo é exibir o número, em decimal, nos *displays* de 7 segmentos e ele sai da ULA um valor em binário de 8 bits, que pode variar de 0 até 255, na base 10. A saída do codificador irá retornar um valor de 12 bits, onde os 4 primeiros bits representam a unidade do número decimal, os 4 segundos representam a dezena e os 4 últimos as centenas, como pode ser visto na Figura 2. Cada um desses 3 valores de 4 bits serão as entradas de cada *display* que recebe um valor, em binário, e exibe seu correspondente em decimal.

3. CONCLUSÃO

Por fim, seguindo todas as orientações presentes neste relatório, espera-se ter um circuito capaz de receber duas entradas de 8 bits cada, *A* e *B*, uma entrada de 4 bits, *S*, que selecionará qual operação será realizada entre *A* e *B*, e exibirá o resultado em três *displays* de 7 segmentos. Além disso, terão dois *flags*, um que representará o *carry*, caso o número resultante ultrapasse os 8 bits, e outro que representará o *zero*, caso o resultado seja 0.

REFERÊNCIAS

Unidade lógica e aritmética. Em: Wikipédia: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Unidade_l%C3%B3gica_e_aritm%C3%A9tica> Acesso em: 2 fev 2021.

VAHID, Frank. **Sistemas digitais: projeto, otimização e HDLS**. Rio Grande do Sul: Artmed Bookman, 2008. 558 p

TOCCI, Ronald J; WIDMER, Neal S; MOSS, Gregory L. **Sistemas digitais: princípios e aplicações**. 11. ed. São Paulo: Pearson, 2011. 817 p

ANEXOS

ANEXO A - Relato Semanal

Líder: Isaac de Lyra Junior

A.1 Equipe

Tabela 1: Identificação da equipe.

Função do grupo:	Discente
Redator	Igor Michael Araujo de Macedo
Debatedor	Eduardo Garcia Zaccharias
Videomaker	Albertho Siziney Costa
Auxiliar	João Matheus Bernardo Rezende

Fonte: Produzido pelos autores.

A.2 Defina o problema

O problema escolhido pelo professor para a semana 03 trata-se de projetar uma Unidade Lógica Aritmética(ULA). A ULA contará com duas entradas de 8 bits, uma entrada de 4 bits e três saídas sendo uma saída de 8 bits e duas de um bit cada. Com relação às entradas, duas serão destinadas para a entrada dos dados a serem operados e uma será utilizada para indicar qual instrução(função) foi selecionada. Com relação às saídas, o circuito deverá apresentar o valor de saída da ULA através dos displays HEX[2:0], a sinalização do bit de carry através do C e a sinalização do bit de Zero através do Z.

A.3 Registro do *brainstorming*

A primeira reunião foi de apresentação, onde foi definido os cargos de cada membro do projeto, entendimento do problema e definição dos objetivos de conceitos a serem estudados. Ficou acordado entre os membros que todos ajudariam no relatório, independentemente de suas respectivas funções.

A segunda reunião foi bem breve onde cada membro mostrou o progresso nos estudos das diferentes funções do ULA, sentimos uma dificuldade em entender como faríamos a função de multiplicação devido o tamanho dos vetores de entrada e saída e como deveria ser usado o carry nas funções que necessitam.

A terceira reunião foi iniciada com o Eduardo mostrando o circuito lógico para as operações de soma, subtração, AND, OR, NOT e XOR implementados no software Proteus, onde ele explicou a lógica dessa implementação para o grupo. A reunião foi continuada pensando em como seriam implementadas as outras funções do ULA de incremento, decremento, multiplicação e deslocamento.

A quarta e última reunião foi de ajustes no relatório e no projeto.

A.4 Pontos-chaves

Um dos pontos chaves é entender como funciona o carry e o borrow que irá carregar a informação para frente ou para trás da soma e subtração de binários, pois a maioria das funções do ULA necessita do entendimento de como deve ser utilizado o carry e o borrow para essa informação não ser perdida.

A.5 Questões de pesquisa

- Código BCD
- Decodificadores e multiplexadores
- Somadores; Deslocadores; Comparadores
- Multiplicadores; Subtratores; Unidade Lógica Aritméticas

A.6 Planejamento da pesquisa

Cronograma de reuniões

HORÁRIOS INDISPONÍVEIS DOS MEMBROS DO GRUPO 4 DE CD - SEMANA 03							
HORÁRIO	SEGUNDA	TERÇA	QUARTA	QUINTA	SEXTA	SÁBADO	DOMINGO
07:00~8:40 (M12)	JOAO	EDUARDO / IGOR	JOAO / EDUARDO	EDUARDO / IGOR	Eduardo	CONCLUIR RELATÓRIO	
8:55~10:35 (M34)		ISAAC /EDUARDO	EDUARDO	ISAAC / Eduardo	eduardo		
10:50~12:30 (M56)		JOAO / IGOR/ EDUARDO	ISAAC / eduardo	JOAO / IGOR/ EDUARDO	ISAAC / eduardo		
13:00~14:40 (T12)		ISAAC / JOAO / IGOR	Segundo Brainstorming(definição da lógica do projeto)	ISAAC / JOAO / IGOR	REUNIÃO 4	CONCLUIR VÍDEO	
14:55~16:35 (T34)		Primeiro Brainstorming e designação dos cargos(todos terão que iniciar o estudo dos conceitos que serão utilizados)	ISAAC /	REUNIÃO 3			
16:50~18:30 (T56)	ISAAC / EDUARDO / IGOR / JOÃO	Eduardo/Albertho	eduardo/Albertho	eduardo/Albertho	eduardo/Albertho	eduardo/Albertho	
18:45~20:25 (N12)	ISAAC / IGOR / JOÃO	Albertho	Albertho	Albertho	Albertho	Albertho	
20:35~22:15 (N34)	eduardo						

A literatura base para pesquisa foi os livros “Sistemas digitais: projeto, otimização e HDLS”(VAHID) e “Sistemas digitais: princípios e aplicações”(TOCCI). Além disso, *papers* e vídeos na plataforma *YouTube.*