Lógica e Sistemas Digitais

Área Departamental de Engenharia de Eletrónica e Telecomunicações e de Computadores

Ricardo Alexandre Ventura Matos	43321	1º Trabalho Prático
Samuel Sampaio Costa	43552	1-11-2016
Pedro Miguel Carvalho Rocha	43884	Grupo 4

Introdução

O presente relatório, referente ao primeiro de três trabalhos práticos, acompanha a realização do projeto de uma ALU[1], de acordo com o enunciado proposto.
Ao longo do relatório descreve-se detalhadamente a realização de cada operação; analisa-se a informação fornecida pelas flags em cada módulo separadamente; dá-se conta da integração dos módulos aritmético e lógico e, por fim, apresentam-se algumas conclusões gerais sobre as funcionalidades da ALU apoiadas nos resultados obtidos.

Operação Adição

Para a realização da operação de adição a 4 bits, foi analisada a forma como se adiciona um bit de cada vez tendo em conta a informação trazida da adição do bit anterior (Carry in) e transportada para a adição do bit seguinte (Carry out). Na prática o Carry corresponde á informação de que a adição de dois bits excedeu o domínio (1 se excedeu e 0 se não excedeu), assim para a adição de 4 bits juntaram-se 4 blocos do operador de adição bit a bit em que o Carry in do bit n+1 é ligado ao Carry out do bit n.
Para a adição bit a bit com Carry temos a seguinte tabela de verdade (tabela 1):

Cn	Bn	An	RAn	Cn+1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabela 1

Cn – Carry In – Carry do bit anterior
Bn – Operando B
An – Operando A
RAn – Resultado da operação Cn+Bn+An
Cn+1 – Carry out – Carry da operação
E assim obtemos as seguintes expressões, usadas na implementação do bloco aritmético:

$$RAn = Cn \oplus An \oplus Bn$$
$$C_{n+1} = An \cdot Cn + An \cdot Bn + Bn \cdot Cn$$

Operação subtração (SBB)

Para a operação de subtração foi reaproveitado o operador adição (técnica denominada *contraction*), com a manipulação de uma entrada, para tal transformou-se uma operação de subtração $R = A - B$ numa soma $R = A + (-B)$, onde o simétrico de B corresponde á soma do seu complemento com 1 (0001) (Tabela 2), ou seja, $R = A + \bar{B} + 1$. Para tal, utilizou-se o modulo BT para transformar o operando B e o modulo C0T para transformar CyBwIn. Essa transformação está dependente da informação de S0 (se S0=0 ADC -> BT e C0T não transformam, se S0=1 SBB -> BT e CoT transformam).

Variável	CyBwOut	Bit 3	Bit 2	Bit 1	Bit 0	Decimal
A		0	1	0	1	5
B		0	1	0	0	4
$\bar{B}$		**1**	**0**	**1**	1	-5
1		0	0	0	1	1
$R = A + \bar{B} + 1$	1	0	0	0	1	1

Tabela 2

[1] A ALU projetada é constituída por dois módulos, interligados conforme o diagrama de blocos no enunciado (fig. 2 e 3 do enunciado). Realiza 4 operações (adição com Carry ADD, subtração com Borrow SBB, Logico AND bit a bit ANL e Shift Right SHR) e fornece adicionalmente 3 flags (Z, CyBwOut e Ov).

Tabela de verdade para bloco BT (tabela 3)

Bn	S_0	BTn
0	0	0
0	1	1
1	0	1
1	1	0

Bn – Operando B
S0 – Seleção (0 -> adição ADC, 1 -> subtração SBB)
BTn – Complemento de B

Tabela 3

Tabela de verdade para bloco C0T (tabela 4)

CyBwi	S_0	CyBwT
0	0	0
0	1	1
1	0	1
1	1	0

CyBwIn – Operando de entrada Carry para ADC ou Borrow para SBB
S0 – Seleção (0 -> adição ADC, 1 -> subtração SBB)
CyBwT – Transformação de CyBwIn que não deve ser alterado para a ADC, e deve ser negado para a SBB

Tabela 4

E assim obtemos as seguintes expressões

$BTn = Bn \oplus S_0$ (implementação do bloco B)
$CyBwT = CyBwIn \oplus S_0$ (implementação do bloco C)

Analise de flags do modulo aritmético

A flag CyBwA corresponde ao Carry do ultimo bit (C4), indicando se o resultado excedeu (1) ou não (0) o domínio em N.

$$CyBwA = C4 \oplus S_0$$

No domino Z existe excesso quando ao adicionarmos dois números positivos e obtemos como resultado um número negativo, ou ao adicionarmos dois números negativos e obtemos como resultado um número positivo[2], pelo que

$$OvA = C3 \oplus C4\ .$$

Operação AND Logico bit a bit (ANL)

A operação AND Lógico bit a bit, corresponde á simples operação AND realizada entre cada bit dos operandos A e B (tabela 5).

	Bit 3	Bit 2	Bit 1	Bit 0
A	0	0	1	1
B	0	1	0	1
A . B	0	0	0	1

An – Bit n do operando A
Bn – Bit n do Operando B
ABn – Bit n da operação An.Bn

Tabela 5

E assim obtemos a seguinte expressão (implementação do bloco ANL):

$ABn = An.Bn$

[2] *In* PARAIZO, José (2008) – *Arquitectura de Computadores*. Cap4- Números e Operadores, p. 4.

Operação Shift Right (SHR)

A operação Shift Right, consiste na translação de cada bit do operando A *n* posições para a direita. O numero de posições que o bit é transladado depende dos valores dos bits 0 e 1 do operando B. Na translação as posições vazias do resultado são preenchidas com o valor de CyBwIn (tabela 6).

		bit 3	bit 2	bit 1	bit 0	
	A	A_3	A_2	A_1	A_0	
$B_{01} = 00$	RSHR	A_3	A_2	A_1	A_0	CyBwOut='0'
$B_{01} = 01$	RSHR	$CyBwIn$	A_3	A_2	A_1	$CyBwOut = A_0$
$B_{01} = 10$	RSHR	$CyBwIn$	$CyBwIn$	A_3	A_2	$CyBwOut = A_1$
$B_{01} = 11$	RSHR	$CyBwIn$	$CyBwIn$	$CyBwIn$	A_3	$CyBwOut = A_2$
		$RSHR_3$	$RSHR_2$	$RSHR_1$	$RSHR_0$	

Tabela 6

Assim cada bit do resultado da operação corresponde a uma seleção de um determinado bit do operando A ou CyBwIn de acordo com os bits de seleção B_{01}. Para isso recorremos a um seletor com 4 bits de entrada, 2 bits de seleção e um resultado. (Multiplexer MUX 4X1)

Tabela de verdade de um Multiplexer MUX 4X1 (tabela 7)

In_0	In_1	In_2	In_3	S_0	S_1	Y
0	--	--	--	0	0	0
1	--	--	--	0	0	1
--	0	--	--	0	1	0
--	1	--	--	0	1	1
--	--	0	--	1	0	0
--	--	1	--	1	0	1
--	--	--	0	1	1	0
--	--	--	1	1	1	1

Tabela 7 – MUX 4x1

$In_{0..3}$ – Valor a selecionar
$S_{0..1}$ – Seletor de 2 bit (4 valores possíveis)
Y - Resultado

E assim obtemos as seguintes expressões:

$$Y = In_0 \cdot \overline{S_0} \cdot \overline{S_1} + In_1 \cdot S_0 \cdot \overline{S_1} + In_2 \cdot \overline{S_0} \cdot S_1 + In_3 \cdot S_0 \cdot S_1$$

Aplicando ao operador temos (implementação do bloco SHR)

$$RSHR_0 = A_0 \cdot \overline{B_0} \cdot \overline{B_1} + A_1 \cdot B_0 \cdot \overline{B_1} + A_2 \cdot \overline{B_0} \cdot B_1 + A_3 \cdot B_0 \cdot B_1$$
$$RSHR_1 = A_1 \cdot \overline{B_0} \cdot \overline{B_1} + A_2 \cdot B_0 \cdot \overline{B_1} + A_3 \cdot \overline{B_0} \cdot B_1 + CyBwIn \cdot B_0 \cdot B_1$$
$$RSHR_2 = A_2 \cdot \overline{B_0} \cdot \overline{B_1} + A_3 \cdot B_0 \cdot \overline{B_1} + CyBwIn \cdot \overline{B_0} \cdot B_1 + CyBwIn \cdot B_0 \cdot B_1$$
$$RSHR_3 = A_3 \cdot \overline{B_0} \cdot \overline{B_1} + CyBwIn \cdot B_0 \cdot \overline{B_1} + CyBwIn \cdot \overline{B_0} \cdot B_1 + CyBwIn \cdot B_0 \cdot B_1$$
$$CyBwo = 0 \cdot \overline{B_0} \cdot \overline{B_1} + A_0 \cdot B_0 \cdot \overline{B_1} + A_1 \cdot \overline{B_0} \cdot B_1 + A_2 \cdot B_0 \cdot B_1$$
$$OvL = A_3 \oplus RSHR_3$$

Integração entre modulo aritmético e logico

Anteriormente analisamos cada uma das operações, agora importa apresentar de uma forma consistente, para tal recorremos a multiplexer MUX 2x1 com a expressão genérica

$$Y = In_0 \cdot \overline{S_0} + In_1 \cdot S_0$$

Que se aplica ao resultado $R_{0..3}$, seleção entre resultados do bloco aritmético $RA_{0..3}$ e bloco logico $RL_{0..3}$, ao CyBwout, seleção entre resultado do bloco aritmético CyBwA e o bloco logico CyBwl, e ao Ov, seleção entre resultado do bloco aritmético OvA e o bloco logico OvR

O Zero obtém-se pela analise do resultado final $R_{0..3}$, em que Z é 1 quando $R_{0..3}$ são 0, e 0 quando não

$$Z = \overline{R0} \cdot \overline{R1} \cdot \overline{R2} \cdot \overline{R3}$$

Conclusões

Depois de programada e montada[3] a ALU, verificou-se que os resultados obtidos estavam em conformidade com o esperado. Seguidamente apresenta-se uma tabela com alguns resultados que constituem ponto de apoio para breves considerações de caráter geral sobre a informação fornecida pela unidade.

Operação	Representação	Inputs A	Inputs B	Ci	Resultado R	Z	de erro Cy/Bw	de erro Ov
ADC	N	11	2	0	13/13	0	0	
	Binário	1011	0010		1101			
	Z	-5	2		-3/-3			0
ADC	N	12	8	1	21/5	0	1	
	Binário	1100	1000		0101			
	Z	-4	-8		-11/5			1
SBB	N	9	11	0	-2/14	0	1	
	Binário	1001	1011		0101			
	Z	-7	-5		-2/-2			0
SBB	N	6	12	0	-6/10	0	1	
	Binário	0110	1100		0101			
	Z	6	-4		10/-6			1
AND	N	7	4	-		0	-	-
	Binário	1011	0101		0001			
	Z						-	-
SHR	N	10	2	0	2/2	0	1	
	Binário	1010	0010		0010			
	Z	-6	2		0/0			1
SHR	N	10	2	1	15	0	0	
	Binário	1100	0010		1111			
	Z	-4	2		-1			0

Tabela 8 - Resultados

Ao realizar uma operação nesta unidade não basta ler diretamente o resultado, é também necessário ter em conta qual foi a operação realizada e a informação fornecida pelas flags. Para a operação adição ADC ou subtração SBB, a flag CyBw ativa indica-nos que o resultado excede o domínio em N (exemplo 3 da tabela 8), e a flag Ov ativa indica-nos que o resultado excede o domínio em Z (exemplo 2 e 4 da tabela 8). Para a operação lógico AND bit a bit ANL, as flags CyBwout e Ov podem em alguns casos ter informação, porém, esta deve ser desprezada. Esta flag pode ativar-se pois o seu valor é calculado pelo bloco Shift Right. Para a operação Shift Right, CyBwOut toma o valor do ultimo bit deslocado, e Ov indica-nos se o valor do quarto bit do resultado é diferente do quarto bit do operando A (exemplos 6 e 7 da tabela 8).

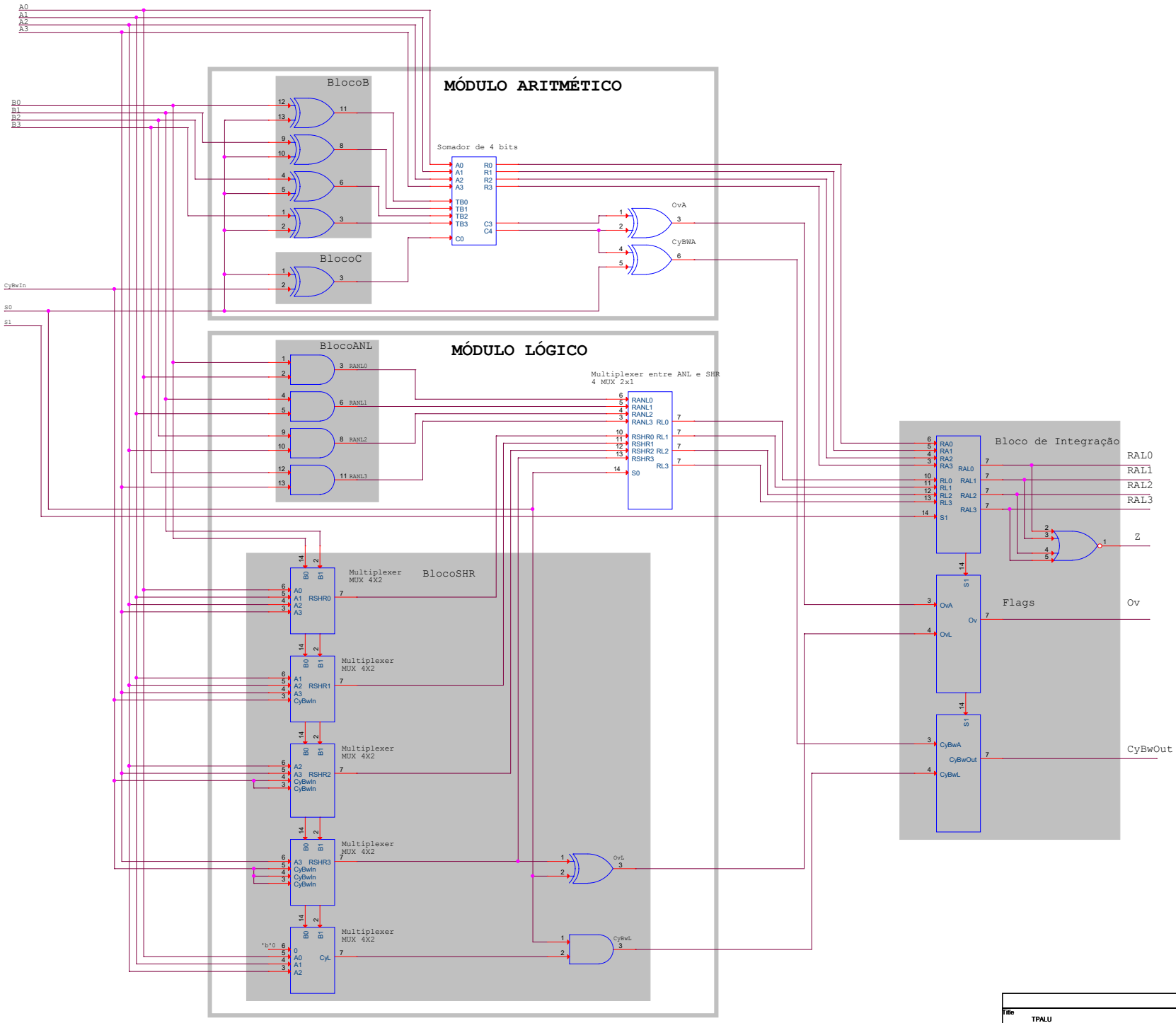
Adicionalmente, podemos aproveitar a operação SBB para relacionar dois números consoante o domínio em que estão expressos. No domínio N, A<B quando a flag CyBw=1 e flag Z=0 (exemplo 3 da tabela 8), A=B quando temos flag CyBw=0 e flag Z=1, A>B quando a flag CyBw=0 e a flag Z=0. No domínio Z, A>B quando (R3=1 e flag Ov=1) ou (R3=0 e flag Ov=0), isto é, R3 igual a Ov, A=B quando flag Z=1, A<B quando R3 diferente de Ov (exemplo 3 da tabela 8).

A0	I1	1	24	5V	VCC
A1	I2	2	23	IO10	C1
A2	I3	3	22	IO9	C2
A3	I4	4	21	IO8	C3
B0	I5	5	20	IO7	RAL0
B1	I6	6	19	IO6	RAL1
B2	I7	7	18	IO5	RAL2
B3	I8	8	17	IO4	RAL3
CyBwIn	I9	9	16	IO3	CyBwOut
S0	I10	10	15	IO2	Z
S1	I11	11	14	IO1	Ov
Massa	GND	12	13	I12	

PAL

Esquema de Ligações da PAL

[3] É fornecido no final do documento um esquema das ligações efetuadas.



Title	TPALU	
Size C	Document Number 01	Rev
Date: Sunday, November 06, 2016	Sheet 1 of 1	