Projeto de ULA

Cris – DRE: XXXXXXXXX
Gus – DRE: XXXXXXXXX
Leo – DRE: XXXXXXXXX

[1]Universidade Federal do Rio de Janeiro (UFRJ)

leonardongc@poli.ufrj.br

1. Enunciado

Fazer ULA de 2 operandos(**A** e **B**) de 4 bits com e 8 operações:

- $A + B$
- $A - B$
- $-A$
- $A + 1$
- Até 2 Oerações de Comparação Bit a Bit
- O Restante das Operações A Escolha do Grupo Utilizando A e B

Com os Flags de **Zero, Negativo, Carry & Overflow**

2. Escolhas

2.1. Lista de Operações Atualizadas

Escolhemos utilizar XOR e ??? pois utilizam submódulos já existentes para outras operações e as operações de deslocamento de bits a direita e a esquerda.

- Soma
- Subtração
- Inversão
- Incremento de 1
- XOR
- ?????
- Right Bitshift
- Left Bitshift

2.2. Projeto do Somador

Separamos a soma em duas etapas, uma que faz a antecipação dos carries e outra que faz a soma bit a bit com os carries.

2.2.1. Antecipador

Calculado Carry a Carry:

$Cin_0 = 0$

$Cin_1 = A_0.B_0$

$Cin_2 = A_1.B_1 + Cin_1(A_1 + B_1) = A_1.B_1 + A_0.B_0(A_2 + B_2)$

$Cin_3 = A_2.B_2 + Cin_2(A_2 + B_2) = A_2.B_2 + A_1.B_1 + A_0.B_0(A_2 + B_2)(A_2 + B_2)$

$Cin_4 = A_3.B_3 + Cin_3(A_3 + B_3) = A_3.B_3 + A_2.B_2 + A_1.B_1 + A_0.B_0(A_2 + B_2)(A_2 + B_2)(A_3 + B_3)$

2.2.2. Somador

Com os carries antecipados é pssível calcular a soma bit a bit:

$F_x = A_x \oplus B_x \oplus Cin_x$

2.3. Projeto do Inversor

O inversor de complemento de 2 tem a seguinte tabela:

A_3	A_2	A_1	A_0	F_3	F_2	F_1	F_0
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

$F_3 = \overline{A_3}.A_2 + \overline{A_3}.A_1 + \overline{A_3}.A_0 + A_3.\overline{A_2}.\overline{A_1}.\overline{A_0}$

$F_2 = \overline{A_2}.A_1 + \overline{A_2}.A_0 + A_2.\overline{A_1}.\overline{A_0}$

$F_1 = \overline{A_1}.A_0 + A_1.\overline{A_0}$

$F_0 = A_0$

2.4. Projeto do Subtrator

Com os módulos de soma e inversão prontos é possível utilizar seus submódulos para construir uma operação de subtração.

2.5. Projeto do Incrmento de 1

Utilizamos o módulo de soma substituindo o operando B pelo vetor $B_3, B_2, B_1 = 0, B_0 = 1$

2.6. Projeto do Comparador XOR

Desconsiderando o submódulo antecipador na soma o submódulo somador vale $A \oplus B$

$$F_x = A_x \oplus B_x$$

2.7. Projeto do Comparador ????

2.8. Projeto do Left Bit Shifter

Montando a Tabela Verdade Para operações de Shift a esquerda é possível verifificar que é certo que não haverá resultado não nulo para $B > 3$, portanto podemos considerar apenas os 2 bits menos significativos do operando B: $B_1 \& B_0$:

B_1	B_0	F_3	F_2	F_1	F_0
0	0	A_3	A_2	A_1	A_0
0	1	A_2	A_1	A_0	0
1	0	A_1	A_0	0	0
1	1	A_0	0	0	0

$$F_3 = \overline{B_1}.\overline{B_0}.A_3 + \overline{B_1}.B_0.A_2 + B_1.\overline{B_0}.A_1 + B_1.B_0.A_0$$

$$F_2 = \overline{B_1}.\overline{B_0}.A_2 + \overline{B_1}.B_0.A_1 + B_1.\overline{B_0}.A_0$$

$$F_1 = \overline{B_1}.\overline{B_0}.A_1 + \overline{B_1}.B_0.A_0$$

$$F_0 = \overline{B_1}.\overline{B_0}.A_0$$

2.9. Projeto do Right Bit Shifter

O Shift a direita por sua vez forma uma tabela similar com as mesmas entradas:

B_1	B_0	F_3	F_2	F_1	F_0
0	0	A_3	A_2	A_1	A_0
0	1	0	A_3	A_2	A_1
1	0	0	0	A_3	A_2
1	1	0	0	0	A_3

$$F_3 = \overline{B_1}.\overline{B_0}.A_3$$

$$F_2 = \overline{B_1}.\overline{B_0}.A_2 + \overline{B_1}.B_0.A_3$$

$$F_1 = \overline{B_1}.\overline{B_0}.A_1 + \overline{B_1}.B_0.A_2 + B_1.\overline{B_0}.A_3$$

$$F_0 = \overline{B_1}.\overline{B_0}.A_0 + \overline{B_1}.B_0.A_1 + B_1.\overline{B_0}.A_2 + B_1.B_0.A_3$$

3. Execução

3.1. Código dos Submódulos

3.1.1. Módulo Antecipador

3.1.2. Módulo Somador

3.1.3. Módulo Inversor

3.1.4. Módulo Shifter

3.2. Implementação dos Flags

3.2.1. Flag de Zero

3.2.2. Flag de Negativo

3.2.3. Flag de Carry

3.2.4. Flag de Overflow