Sistemas Digitais

Somador de Ponto Flutuante

Profª. José Artur Quilci

Lucas Thiago Zane, 11060515

Daniel Vieira Batista, 11106614

Código fonte disponível em https://github.com/lucaszane/SomadorPontoFlutuante/

Santo André

2019

Introdução

Este projeto é um somador de ponto flutuante. Sua função é realizar uma das operações mais básicas de um computador comum, a soma. Na realização deste projeto, são abordados temas de suma importância para a computação, como transformação de base numérica, mantissa, soma binária, entre outros. Estes pontos são fundamentais para a computação atualmente e são a partir deles que se derivam toda a variedade de coisas que a computação consegue alcançar.

Objetivos

Este projeto tem como objetivo realizar o desenvolvimento de um somador de ponto flutuante, onde dadas duas entradas, obtenha-se uma saída contendo a soma das entradas no display seven segments da placa DE1 Altera disponibilizada pela Universidade Federal do ABC para os alunos do curso de Sistemas Digitais.

O objetivo inicial era obter entradas em números fracionários na base binária da seguinte forma:

+/- 0,00000000 x 2^0000

Onde o lado direito da vírgula representa o número em si e o expoente representa a ordem de grandeza do número. Dado que a placa DE1 Altera não possui 26 entradas (13 para cada número da soma, onde 1 é para o sinal, 8 para a parte fracionária e 4 para o expoente), os projetistas deveriam encontrar uma maneira adequada de ler estas entradas e realizar a soma de forma correta, exibindo no display Seven Segments da seguinte forma:

-	0	0	0

Ou seja, o display contaria com 4 dígitos, o primeiro conteria o sinal (negativo ou vazio, que representa o positivo), o segundo e o terceiro display conteriam, na base Hexadecimal, a primeira e a segunda parte da fração (0 ~ F na base hexadecimal que representa 0000 ~ 1111 na base binária), respectivamente, com 4 bits cada um e o quarto display contendo o expoente, também na base Hexadecimal.

Metodologia

Este trabalho foi realizado utilizando-se de simulações no GTKWave e testes realizados na placa DE1 Altera. Para a leitura das entradas foi criado uma máquina de estados com 4 estados, vistos na Figura 6 e explicados aqui, sendo que a leitura foi feita utilizando as chaves SW da placa:

- Estado 1 para a leitura do sinal 1 e fração 1;
- Estado 2 para a leitura do expoente 1;
- Estado 3 para a leitura do sinal 2 e fração 2;
- Estado 4 para a leitura do expoente 2;

Após a leitura das entradas de cada estado, é possível verificar o resultado da soma no display Seven Segments que foi feito utilizando os arquivos auxiliares “hex_to_sseg.vhd” e “disp_mux.vhd”, de forma que o primeiro realiza a conversão dos números obtidos para o equivalente no display e o segundo escolhe qual valor deve ser exibido em cada um dos 4 seven segment displays.

A simulação do projeto foi feita no GTKWave e o teste realizado com a placa DE1 Altera utilizando a ferramenta Quartus II para compilação e carga do projeto na placa.

Apresentação dos Dados

1 - Foi feita a execução do somador de ponto flutuante no **GTKWave,** tendo o seu resultado apresentado na Figura 1.

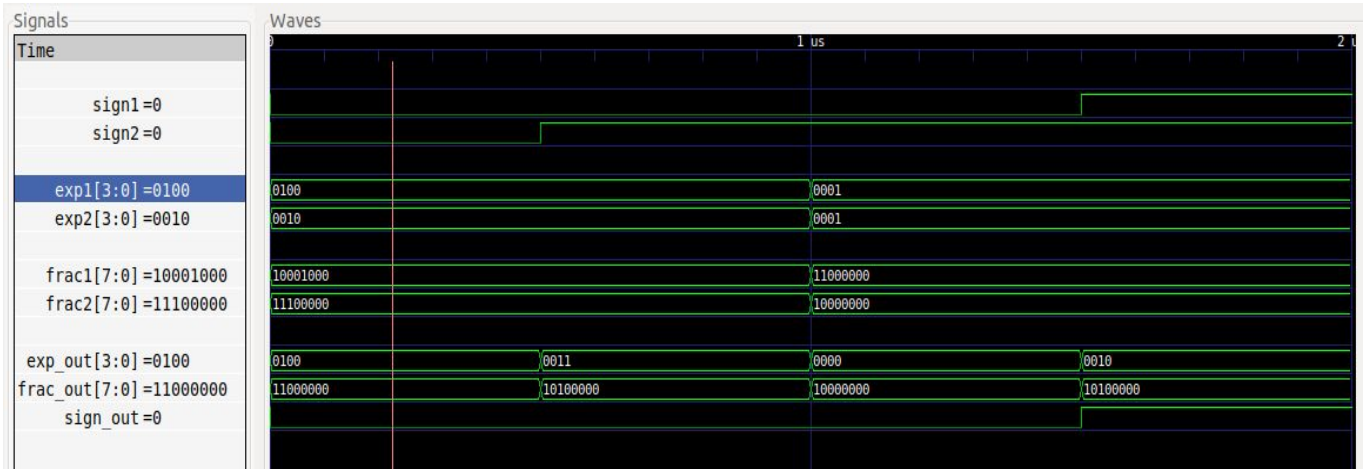

Figura 1 - Execução do projeto com testbench modificado

O testbench foi alterado para realizar o teste com quatro entradas diferentes, obtendo os resultados da Figura 1 e para fins de simplicidade, destacados em decimal na Tabela 1.

Clock	Primeiro número (decimal)	Segundo Número (decimal)	Resultado (decimal)
1	+8.5	+3.5	+12.0
2	+8.5	-3.5	+5
3	+1.5	-1.0	+0.5
4	-1.5	-1.0	-2.5

Tabela 1 - Entradas e saídas do somador

Para todas as operações, colocamos o número maior em cima, alinharmos os expoentes dos binários e então fazemos a operação e no fim, normalizamos novamente.

Para a operação no 1º Clock, temos:

- sign1 = 0, exp1 = 0100, frac1 = 10001000;
- sign2 = 0, exp2 = 0010, frac2 = 11100000;
- Então, alinhamos o exp2 para 0100 e frac2 para 00111000;

```
  10001000
+ 00111000
  11000000
```

- Como o resultado já está normalizado, não precisamos mais mexer no expoente, que se manterá 0100.

Para a operação no 2º Clock, temos:

- sign1 = 0, exp1 = 0100, frac1 = 10001000;
- sign2 = 1, exp2 = 0010, frac2 = 11100000;
- Então, alinhamos o exp2 para 0100 e frac2 para 00111000;

```
  10001000
- 00111000
  101010000
```

- Como temos um bit “carry”, removemos ele e fazemos o shift para a esquerda resultando em 01010000 e como ainda temos um zero à esquerda, fazemos um outro shift alterando o expoente para 0011 e a fração resultando em 10100000.

Para a operação no 3º Clock, temos:

- sign1 = 0, exp1 = 0001, frac1 = 11000000;
- sign2 = 1, exp2 = 0001, frac2 = 10000000;

11000000
- 10000000
01000000

- Como exp1 e exp2 estão alinhados e possuem valor 1: 0,1 binário = 0.5 decimal;

Para a operação no 4º Clock, temos:

- sign1 = 1, exp1 = 0001, frac1 =11000000;
- sign2 = 1, exp2 = 0001, frac2 = 10000000;
-

- 11000000
- 10000000
101000000

- Como temos um bit 0 “carry” logo o desprezamos e “shiftamos” para a esquerda: -1,01000000, normalizando -10,1000000 binário = -2,5 decimal.

2 - Foram feitos testes na placa DE1 Altera com diversas entradas, obtendo os seguintes resultados para cada entrada:

1) $8{,}5_{10} + 3_{10} = 11{,}5_{10} = 1011{,}1000_2$

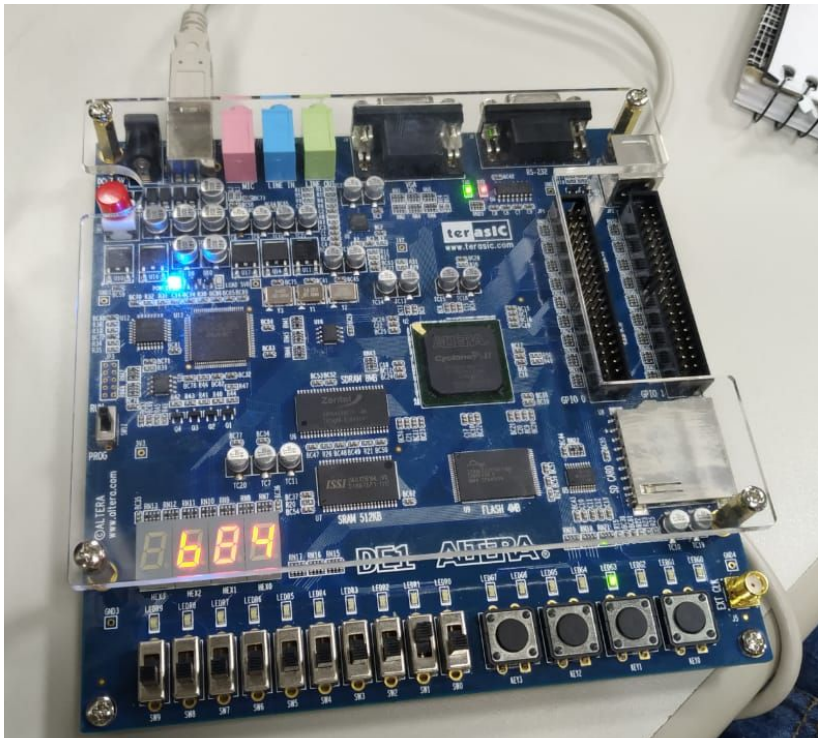

Figura 2 - Display do somador. Primeiro teste.

2) $8,5_{10} - 3,5_{10} = 5_{10} = 0101,0000_2$

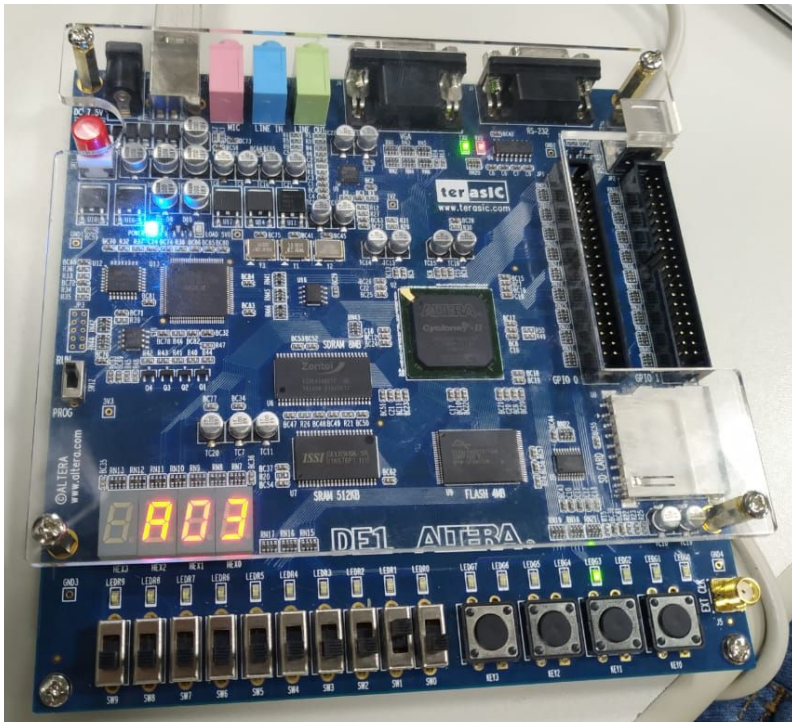

Figura 3 - Display do somador. Segundo teste.

3) $-1,5_{10} - 1,25_{10} = -2,75_{10} = - 0010,1100_2$

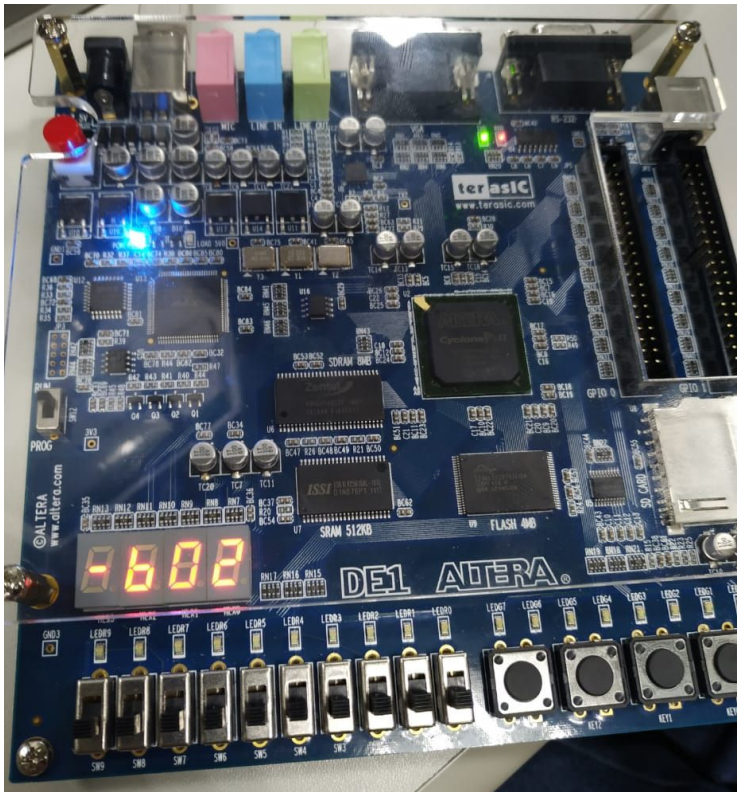

Figura 4 - Display do somador. Terceiro Teste

Mini-Tutorial

O projeto é composto por quatro componentes: display de operações, entrada de dados numéricos, botões de estado e leds de indicação. O display de operações mostra os resultados e modificações realizadas nos estados e nas entradas. As entradas de dados recebem os bits que representam a mantissa dos números e seus expoentes. O projeto também contempla os botões de estados que mudam os estados internos de armazenamento e que podem ser acompanhados pelas leds de indicação.

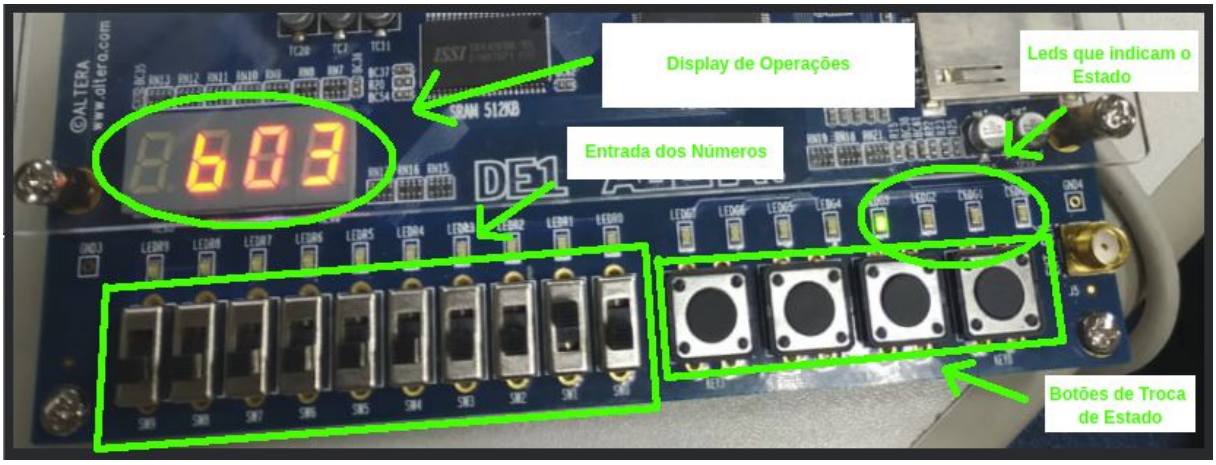

Figura 5 - Descrição da placa

O projeto se baseia em 4 estados para realizar a operação. Para mudar de estado devemos pressionar os botões de estados, cada botão representa um estado específico: KEY0 indica o Estado 1, KEY1 indica o Estado 2, KEY2 indica o Estado 3 e KEY4 indica o Estado 4. Após especificar o estado escolhido, devemos indicar em SW (de SW0 até SW7) a mantissa do número específico e seu sinal em SW8, no próximo estado entramos com expoente do número do estado anterior. Segue-se nesse processo até o quarto estado, onde a operação estará finalizada e pode ser vista no display. Uma representação esquemática pode ser vista na Figura 6.

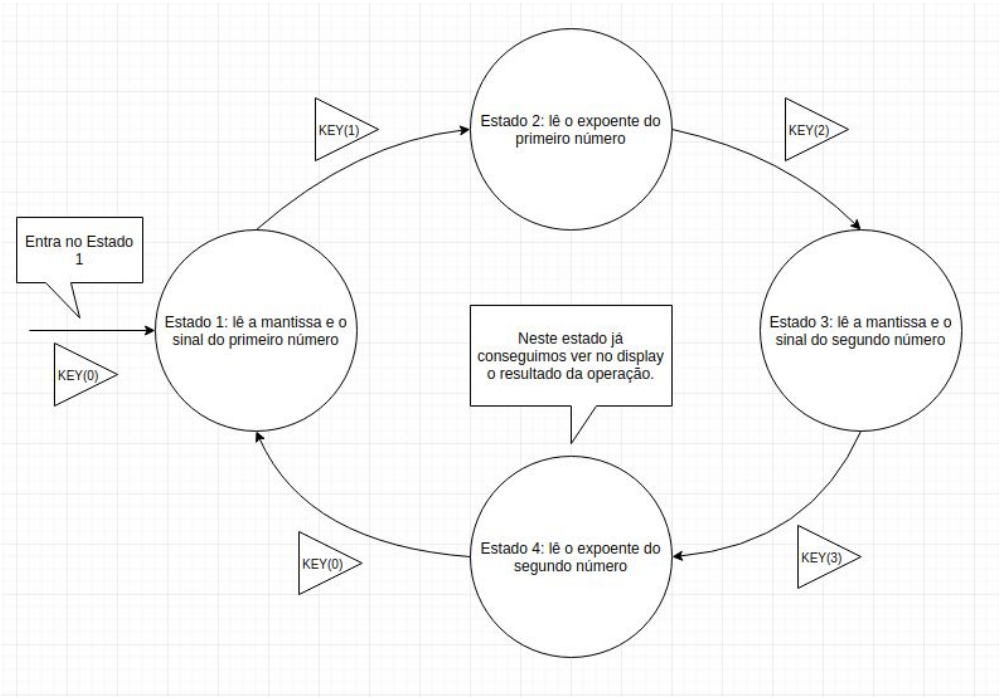

Figura 6 - Máquina de Estados

O diagrama da figura 6 funciona indica as seguintes operações:

- Para entrar no Estado 1, é pressionado o primeiro botão KEY0 e a LED0 será acesa;
- O display será zerado e deve ser especificado em SW7 (MSB) até SW0 (LSB) a mantissa do primeiro número binário e seu sinal em SW8, após a especificação o botão de KEY1 deve ser pressionado para o Estado 2;
- No Estado 2 (LED1 aceso) o expoente do primeiro número binário deve ser especificado em SW3 (MSB) até SW0 (LSB). Após isso, o botão KEY2 é pressionado e leva a máquina para o Estado 3;
- Estando no Estado 3 (LED2 aceso) deve ser especificado em SW7 (MSB) até SW0 (LSB) a mantissa do segundo número binário e seu sinal em SW8, após a especificação o botão de KEY3 deve ser pressionado para o Estado 4;
- No Estado 4 (LED3 aceso) o expoente do segundo número binário deve ser especificado em SW3 (MSB) até SW0 (LSB). Após isso, o valor disposto no display será o resultado da operação. Na figura 7 temos um exemplo de resultado.

- Para zerar todas as entradas, basta percorrer por todos os estados com as chaves SW no 0.

Abaixo observamos o resultado de uma operação exemplo. O primeiro display de sete segmentos indica o sinal da operação, o segundo e o terceiro indicam um número em hexadecimal e o quarto mostra o expoente final em hexadecimal. Para converter o resultado no valor final devemos escrever o valor disposto no HEX2 em binário e o valor disposto em HEX1 em binário e concatená-los, devemos considerar o resultado da concatenação como um ponto flutuante deslocando a vírgula para a direita na quantidade de vezes especificada no display HEX0.

No Figura temos o seguinte número: - 0, 1000 0100 x 2^1 = -1.000100 em binário = 1,0625 em decimal.

Figura 7 - Resultado exibido no Seven Segment Display

Conclusão

Os objetivos deste projeto foram alcançados através das simulações e dos testes realizados durante a elaboração do projeto. Utilizando a metodologia, o projeto foi concluído de forma a ler adequadamente as entradas de acordo com a máquina de estados proposta, realizar a operação de soma e exibir o resultado no Seven Segment Display. Com tudo isto feito, foi possível obter o resultado esperado para o projeto, utilizando todos os arquivos e recursos disponibilizados para a turma.