SISTEMAS DIGITAIS

Somador de Ponto Flutuante

Professor José Artur Quilici

Jairo da Silva Freitas Júnior 11019214
Vinícius Soares Xavier 11060812

Santo André
2019

1. Introdução

O campo da eletrônica digital revolucionou o mundo ao automatizar parte das atividades humanas, caracterizando-se como uma alavanca para qualidade de vida da população mundial. Apenas o mercado global de eletrônicos destinados ao consumidor final estava avaliado em US$ 1,17 trilhões no final de 2017 [1].

A grande demanda por aplicações de chips eletrônicos versáteis e em volumes de produção mais baixos, assim como o barateamento dos transistores (pela Lei de Moore) provocaram o desenvolvimento das FPGAs (Field Programmable Gate Arrays) em 1983 por Ross Freeman (co-fundador da Xilinx Inc., atual líder de mercado cujo valor das ações em NASDAQ cresceu 77 vezes desde 1990) [2]. FPGAs são dispositivos físicos que incorporam a implementação programática de circuitos lógicos. A maior vantagem destas placas é permitir a prototipação de circuitos com baixo custo e em hardware reutilizável.

O projeto VHDL produzido neste trabalho pode ser encontrado em:
https://github.com/vinxavier/SomadorFlutuante

2. Objetivos

O objetivo geral deste projeto é implementar um somador de ponto flutuante na FPGA DE1 da Altera. Os objetivos específicos foram:

- Adaptar e compilar os códigos fonte fornecidos na atividade avaliativa.
- Criar um test bench para simulação do funcionamento do somador de ponto flutuante no GTKWake e intepretar o resultado.
- Superar a especificação proposta na atividade implementando o somador de ponto flutuante como uma máquina de estados finita, permitindo assim maior número de entradas no projeto.
- Simular o circuito em uma FPGA DE1 da Altera usando o Quartus II/ModelSim.

3. Justificativa

O trabalho foi desenvolvido como atividade de conclusão da disciplina Sistemas Digitais (MCTA024) da Universidade Federal do ABC, com a finalidade didática de consolidar o conteúdo teórico através de projeto prático. Em última instância, este projeto justifica-se como parte essencial na formação de futuros projetistas de sistemas digitais.

4. Metodologia

A implementação do somador de ponto flutuante é uma adaptação da solução proposta por Pong Chu [3]. O algoritmo consiste em quatro passos:

1. **Ordenar** os números de acordo com o maior valor em módulo.
2. **Alinhar** os números para que tenham o mesmo expoente. O número com expoente menor tem seu expoente incrementado. Para cada incremento, a parte fracionária do menor número é deslocada uma casa para a direita (em direção aos dígitos menos significativos).
3. **Adição/subtração**: Adiciona bit a bit a parte fracionária dos dois números.
4. **Normalização**:
    a. Caso haja carry out na soma, incrementar o expoente.
    b. Após uma subtração, se houverem zeros a esquerda, reduzir o expoente e deslocar a parte fracionária para a esquerda até que o primeiro dígito seja 1.
    c. Caso após uma subtração o valor seja muito pequeno para ser normalizado, substituir por zero.

Embora o mesmo algoritmo tenha sido utilizado, a implementação de hardware de Pong Chu permite selecionar apenas alguns bits de cada fator da soma. Este trabalho permite selecionar números positivos ou negativos com expoente de 0 a 15 e parte fracionária de 0,5 a 0,996 (255/256). Em suma, valores no intervalo entre -5.0E14 e 1.0E15 podem ser representados.

A simulação realizada no GTKWake foi do tipo funcional com resultados que não levam em consideração atrasos e tempo de propagação dos sinais. Ou seja, dadas as entradas, o resultado é exibido no clock seguinte.

A configuração do projeto na FPGA foi feita como uma máquina de estados finitos com 7 estados. A figura 0 ilustra a máquina. Em seguida descreve-se cada estado em detalhe.

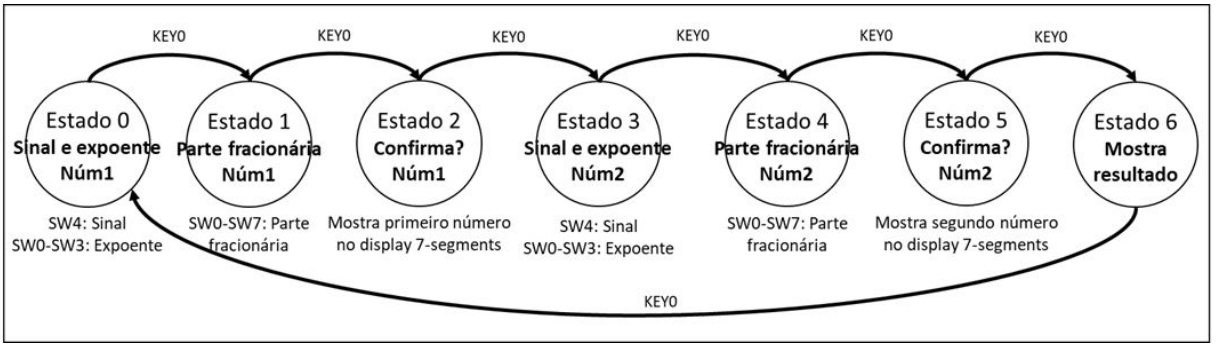

Figura 0: Máquina de estados finitos implementada

●**Estado 0** : O programa inicia mostrando zero no display de seven segment como é observado na Figura 1. Utilize SW4 para dizer o sinal do número 1 e SW0 a SW3 para definir o expoente do número 1. Pressione KEY0 para confirmar o sinal e o expoente e seguir para o próximo estado. É possível ver no display o sinal e expoente inseridos.

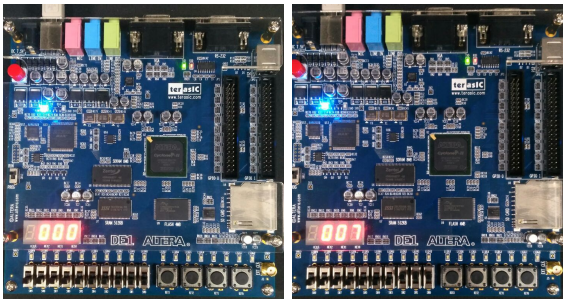

Figura 1 - Estado inicial. A esquerda o programa iniciado. A direita a chave SW4 desligada e as chaves SW0-SW3 formando o número 7 representando o expoente que é possível ver no display.

●**Estado 1:** Utilize SW0 a SW7 para definir a parte fracionária do número 1, e aperte KEY0 para confirmar e ir para o próximo estado.

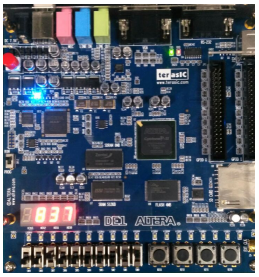

Figura 2 - As chaves SW0-SW7 definem a parte fracionária que é mostrada no HEX1 e no HEX2.

● **Estado 2**: Mostra a Entrada 1 na tela. Pressione KEY0 para confirmar e ir para o próximo estado. Pressione KEY1 para reiniciar

● **Estado 3**: Utilize SW4 para inserir o sinal do número 2 e SW0 a SW3 para definir o expoente do número 2. Pressione KEY0 para confirmar o sinal e o expoente e ir para o próximo estado. É possível ver no display (conforme Figura 3) o sinal e expoente desse número.

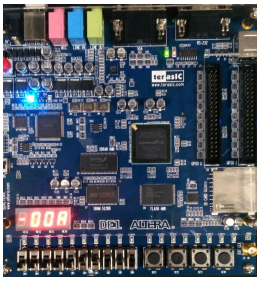

Figura 3 - A chave SW4 acionada mostra o sinal negativo, e é possível ver que a potência escolhida para o número dois é A(10 em decimal).

● **Estado 4**: Utilize SW0 a SW7 para definir a parte fracionária do número 2, e aperte KEY0 para confirmar e ir para o próximo estado. É possível ver no display o sinal e expoente desse número (conforme Figura 4).

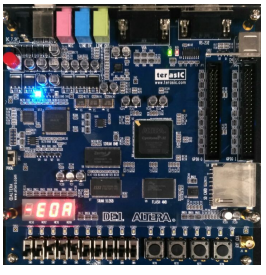

Figura 4 - Informando a parte fracionária do número 2.

● **Estado 5**: Mostra a Entrada 2 na tela. Pressione KEY0 para confirmar e ir para o próximo estado.

● **Estado 6**: É o estado resposta. Neste estado o resultado será exibido no 7 Segment display e nos LEDs. Ao apertar KEY0, volta ao estado 0. Os LEDs vermelhos de 0 até 3 respresentam o expoente, o LED vermelho 4 o sinal e os LEDs verdes representam a parte fracionária de 8 bits.

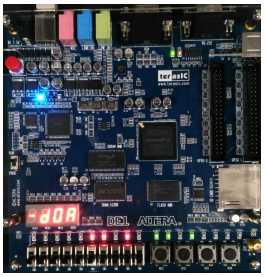

Figura 5 - Exibição do resultado na tela com a confirmação com os LEDS vermelhos e verdes.

Ao apertar KEY1 cancela a operação e volta ao estado 0.

5. Apresentação de dados

Foram realizados 4 testes no GHDL. A Figura 6 é um print dos resultados obtidos. SignN é o sinal do número (0 = positivo e 1=negativo), expN é o expoente e fracN a parte fracionária, sendo que N={1, 2, _out}, onde _out refere-se a saída da soma. A Tabela 1 mostra com mais resolução cada teste executado.

Conforme pode-se notar no teste 4, a adição de um número muito grande a outro muito pequeno “estoura” a resolução do somador, mantendo apenas a parte mais significativa do resultado. Os mesmos testes foram executados na FPGA e os resultados foram idênticos.

Figura 6 - Resultados no GHDL

Teste #	Número	Sinal	Expoente	Francionária
1	1	0	0111	10000011
1	2	0	1010	11100000

1	**Saída (1+2)**	**0**	**1010**	**11110000**
2	3	0	0111	10000011
2	4	1	1010	11100000
2	**Saída (3+4)**	**1**	**1010**	**11010000**
3	5	1	0111	10000011
3	6	1	1010	11100000
3	**Saída (5+6)**	**1**	**1010**	**11110000**
4	7	0	1111	10000000
4	8	0	0000	10000000
4	**Saída (5+6)**	**0**	**1111**	**10000000**

Tabela 1: Resultados no GHDL

6. Conclusão

Neste trabalho foi possível desenvolver em VHDL um somador de ponto flutuante com implementação na forma de máquina de estados finitos em uma FPGA. Os resultados dos testes em hardware e simulação foram consistentes com os cálculos manuais e discussões sobre a limitação de representação numérica da solução foram discutidos.

7. Referências

[1] Zion Market Research. **Consumer Electronics Market by Product (Smartphone, Television, DVD players, Refrigerators, Washing machines, Digital cameras, and Hard disk drives) and by Region (North America, Europe, Asia Pacific, Latin America, and the Middle East & Africa) - Global Industry Perspective, Comprehensive Analysis and Forecast, 2017 – 2024**. 110p. 2018. Disponível em: https://www.zionmarketresearch.com/report/consumer-electronics-market. Acesso em: 27/08/2019

[2] Ross Freeman. Wikipedia. Disponível em: https://en.wikipedia.org/wiki/Ross_Freeman. Acesso em: 27/08/2019

[3] Pong P. Chu. **FPGA Prototyping by VHDL Examples.** Hoboken, NJ: John Wiley & Sons, inc., 2008.