IMPLEMENTAÇÃO DE CIRCUITOS ARITMÉTICOS EM PONTO FLUTUANTE, UTILIZANDO FORMATO COM NÚMERO DE BITS CONFIGURÁVEL

VICTOR A. M. SANTOS, EDER B. KAPISCH, LEANDRO R. M. SILVA, LUCIANO M. DE A. FILHO

Departamento de Circuitos Elétricos, Universidade Federal de Juiz de Fora - UFJF
RUA JOSÉ LOURENÇO KELMER, S/N - SÃO PEDRO, JUIZ DE FORA - MG
E-MAILS: VICTOR.AUGUSTO@ENGENHARIA.UFJF.BR, LEANDRO.MANSO@UFJF.EDU.BR,
LUCIANO.MANHAES@UFJF.EDU.BR

Abstract— A representation of real numbers in binary systems caused floating-point numeric representations to be established. In addition to accurately representing a numeric value, calculating the result of arithmetic operations between two floating-point values, such as adding, multiplying, dividing, and subtracting, without error and with immediate response time is a crucial point. Currently, the IEEE 754 standard is the representation pattern used in computational systems, and this work presents a simplified floating-point representation implementation, in which the number of bits used is parametrizable. The implemented circuits only use combinational logic, an aim to optimize the response time of the operations, and a number of logical elements, according to a desired application.

Keywords— Floating-point representation, IEEE 754, arithmetic circuits, parameterizable circuits.

Resumo— A representação de números reais em sistemas binários fez com que fossem estabelecidas representações numéricas em ponto flutuante. Além de representar precisamente um valor numérico, calcular o resultado de operações aritméticas entre dois valores em ponto flutuante, como somar, multiplicar, dividir e subtrair, sem que haja erro e com tempo de resposta imediato é um ponto crucial. Atualmente o padrão IEEE 754 é o padrão de representação utilizado nos sistemas computacionais, e este trabalho apresenta uma implementação de uma representação em ponto flutuante simplificada, em que o número de bits utilizado é parametrizável. Os circuitos implementados fazem uso apenas de lógica combinacional, a fim de otimizar o tempo de resposta das operações, e a quantidade de elementos lógicos utilizada, de acordo com a aplicação desejada.

Palavras-chave— Representação em ponto flutuante, IEEE 754, circuitos aritméticos, circuitos parametrizáveis.

1 Introdução

A representação na base binária utilizando o formato de Ponto Flutuante, surgiu da necessidade de se representar números reais em sistemas digitais. Nesse contexto, a aritmética de ponto flutuantes é uma maneira de operar os números reais de modo aproximado, mantendo um *trade-off* entre o alcance e a precisão da representação.

Dessa forma, as operações em ponto flutuante são frequentemente encontradas em sistemas operam com números em uma grande faixa (números muito grandes e muito pequenos), que exigem tempos de processamento rápidos. Um número é, em geral, representado aproximadamente para um número fixo de dígitos significativos, e escalado usando um expoente em alguma base fixa. Em sistemas digitais, a base utilizada é a base binária.

O termo ponto flutuante refere-se ao fato de que o ponto pode "flutuar", isto é, pode ser colocado em qualquer lugar em relação aos dígitos significativos do número. Esta posição é indicada como o componente exponente e, portanto, a representação de ponto flutuante pode ser considerada como uma espécie de notação científica.

Ao longo dos anos, grande variedade de representações de ponto flutuante foram utilizadas em computadores. No entanto, desde a década de 1990, a representação mais comumente encontrada é a definida pelo padrão IEEE 754 (Singh, Prateek & Bhole, 2014).

Por exemplo, as FPGAs, Field Programmable Gate Array, em português "Arranjo de Portas Programáveis em Campo", são amplamente utilizadas para computação científica por que tem facilidade de sintetizar um hardware customizável para a aplicação. Com isso, sabe-se que o ponto flutuante também é importante para cálculos científicos para sua estabilidade numérica. Porém, apesar da fraca performance em ponto flutuante das FPGAs, houve um interesse significativo em usar FPGAs para aplicações científicas.

Enquanto os FPGAs se destacam em aplicações e ponto fixo, aplicações de ponto flutuante ocupam uma quantidade grande e muitas vezes impraticável de recursos quando implementado nesses dispositivos. O tamanho e a arquitetura limitados de FPGAs não são adequados para aplicações de ponto flutuante (Dou, et al., 2005; Hemmert & Underwood, 2005; Govindu, et al., 2004; DeLorimier & DeHon, 2005)

Daí, a proposta de uma representação em ponto flutuante que seja mais rápida, precisa, e que utilize menos recursos das FPGAs, por exemplo, é um ponto crucial de pesquisa deste trabalho.

Com isso, a velocidade das operações de ponto flutuante, é uma característica importante de um sistema digital, especialmente para aplicações que envolvam cálculos matemáticos intensivos, em que a otimização desse processo de aritmética é a motivação para que pesquisas sejam realizadas.

O presente trabalho tem como objetivos: apresentar um novo formato para representação de número em ponto flutuante, que seja parametrizável em relação ao número de bits; além de descrever circuitos aritméti-

cos capazes de realizar as operações de soma/subtração, multiplicação e divisão, de maneira combinacional, utilizando esse formato; e por fim, comparar os resultados dessas operações com os resultados obtidos utilizando o padrão IEEE 754.

Portanto, o mesmo está dividido da seguinte maneira: a Seção 2 apresenta o formato de ponto flutuante estabelecido pelo padrão IEEE 754; a Seção 3 será apresentado o formato de ponto flutuante proposto, bem como serão descritas as operações aritméticas utilizando esse formato; os resultados serão mostrados na Seção 4; e por fim, a Seção 5 apresenta a conclusão do trabalho.

2 Formato Ponto Flutuante

De maneira a generalizar as operações com números em formato de ponto flutuante para vários hardwares distintos, é necessário a adoção de um padrão para a representação desses números na base binária, já que é a base com a qual os sistemas digitais trabalham.

O padrão adotado atualmente para a representação de números em ponto flutuante é o IEEE 754, que suporta dois formatos, o *single* e o *double*, que utilizam 32 e 64 bits, respectivamente.

2.1 Representação de um Ponto Flutuante

A representação em binário de um número em ponto flutuante é semelhante a representação de números reais em notação científica, e possui a seguinte forma:

$$(-1)^S \; x \; M(2)^{\pm E} \qquad (1)$$

em que, S representa o sinal do número (S=0 para números positivos e S=1 para números negativos), M a mantissa e E o expoente.

Um problema dessa representação (notação científica), é que um mesmo número pode ser representado de várias maneiras. Por exemplo, os seguintes números 1, 01×2^2; 10, 1×2^1; 0, 0101×2^4, são equivalentes. Para simplificar a forma de representação em ponto flutuante, o formato IEEE 754 determina que o número seja representado de maneira normalizada. A normalização é o deslocamento do ponto para esquerda ou para direita, de forma que o mesmo fique à direita do dígito diferente de zero mais à esquerda. Utilizando do exemplo anterior, o valor normalizado terá a seguinte forma: $1, 01 \times 2^2$

Quando se aplica a normalização, o primeiro dígito à esquerda do ponto será sempre 1, esse dígito não terá mais necessidade de ser armazenado, sendo denominado, dígito normalizado (David & John, 2005).

2.2 Padrão IEEE 754

O padrão IEEE 754 foi estabelecido para criar uma normalização na representação de números em ponto flutuante na base binária, assim tornando mais compatível o uso entre um programa em várias hardwares distintos. Esse padrão, teve por diretrizes a portabilidade de aplicações já existentes, entre diversos computadores com esse padrão, de tal forma que apresentem o mesmo resultado; além de tornar fácil e seguro produzir programas para a área matemática, promovendo assim funções elementares que seguem um padrão.

O IEEE 754, especifica formatos *single* e *double precision* para números em ponto flutuante, operações de adição, subtração, multiplicação, divisão, raiz quadrada, resto e comparação, conversões entre inteiros e ponto flutuante, funções de conversão entre diferentes formatos de ponto flutuante, funções de conversão entre números de ponto flutuante no formato básico para cadeia de caracteres (strings) decimais, exceções de ponto flutuante e sua manipulação, incluindo NaN (*Not a Numbers*).

O padrão IEEE-754 interpreta um número em ponto flutuante binário como se o mesmo fosse dividido em quatro grupos: sinal, expoente, bit implícito, parte fracionária (mantissa).

1. O sinal indica se o número é negativo (1) ou positivo (0);
2. O expoente armazena a potência de 2 para a qual o número será elevado, o mesmo é baseado em uma representação transladada ou polarizada (bias exponent). Isto significa que o valor do expoente do número em ponto flutuante é representado por um número inteiro mais um deslocamento (bias);
3. O bit implícito, ou seja, normalizado, fica à esquerda da parte fracionária. Em regra, este valor será 1 (um) e, portanto, não será necessário seu armazenamento;
4. A parte fracionária é o valor que fica à direita do ponto flutuante. Denomina-se, mantissa, a junção do bit implícito mais a parte fracionária;

O formato de ponto flutuante de precisão simples (32 bits) consiste, em um bit de sinal (S), 8 bits de expoente (E), e 23 bits de mantissa (M). O campo de expoente (E), corresponde à soma de 127 com o expoente na base 2 do número representado. O campo de mantissa (M), corresponde à parte fracionária da mantissa do número representado.

Considera-se sempre a mantissa normalizada entre 1 e 2. Desta forma, a sua parte inteira é sempre um bit igual a 1(um) que não é necessário representar. A representação do ponto flutuante para 32 bits de acordo com o IEEE, é descrita como:

$$(-1)^S \; x \; 1.M(2)^{E-127} \qquad (2)$$

em que, S é o sinal, M é a mantissa, 2 é a base, que neste caso é binária, e E é o expoente.

Além dos números discutidos acima, o IEEE-754 inclui valores especiais representando NaN (Not a Number), $+\infty$ e $-\infty$. Valores especiais são codificados com o uso do expoente fora da faixa de máximo. Os valores -0.0 e +0.0 são distinguíveis embora seja verdadeira a expressão +0.0 = -0.0.

Os NaNs são usados no lugar de vários resultados inválidos; tal como por exemplo a indeterminação "0/0". Na realidade muitos padrões diferentes de bits podem codificar um valor NaN. A intenção é permitir que sejam repassadas informações adicionais no significando do NaN, tais como o endereço da operação inválida que gerou um NaN. As informações contidas nesses NaNs só podem ser obtidas por meios não aritméticos.

Para atribuição e operações aritméticas, se um único NaN é dado como um operando, o mesmo NaN deverá ser retornado como resultado; se dois operandos NaN forem fornecidos, um deles deverá ser retornado.

O padrão também define a comparação entre números em ponto flutuante, como por exemplo, as relações, maior que (>), igual (=) e menor que (<). O padrão também estabelece que não se pode comparar NaNs com as relações primitivas. Um NaN não é nem menor que, nem maior que, nem igual a qualquer valor em ponto flutuante, inclusive ele mesmo. Portanto, as expressões NaN > *a*, NaN = *a*, NaN < *a* (sendo *a* um valor), são todas falsas, para qualquer valor, inclusive *a* sendo um outro NaN.

3 Formato de Ponto Flutuante Proposto

Nesta seção será descrito o formato de ponto flutuante proposto, bem como as operações aritméticas utilizando-o. Em que a proposta é desenvolver circuitos digitais puramente combinacionais e parametrizáveis em relação ao número de bits, diferentemente da maioria dos circuitos que se utilizam do padrão IEEE 754, que operam em pipeline e, portanto, necessitam de alguns ciclos de clock para fornecer o resultado e com número de bits fixo.

A proposta de um ponto flutuante simplificado, têm sua estrutura de representação semelhante à do formato IEEE 754, em que o número é dividido em sinal, mantissa e expoente. O formato proposto será apresentado utilizando 32 bits, porém, vale a pena frisar que, como o foco é a utilização desse formato em circuitos sintetizados em FPGA, qualquer quantidade de bits pode ser utilizada.

3.1.1 Formato

O formato proposto, interpreta um número em ponto flutuante binário, semelhante ao padrão IEEE 754, como se o mesmo fosse dividido em quatro grupos: sinal, expoente, bit implícito, mantissa.

Em relação a representação binária, o formato de representação em ponto flutuante proposto, realiza em seu expoente o complemento de 2 do número, e na mantissa, neste caso chamada de mantissa normalizada, pois o bit mais significativo deve ser sempre 1, fazendo com o que a representação seja aceita.

O formato de ponto flutuante proposto de precisão simples (32 bits) consiste, num bit de sinal (S), 8 bits de expoente (E), e uma mantissa de 23 bits (M), normalizada. O campo de mantissa (M), corresponde a parte inteira e a parte fracionária da mantissa do número representado. Considera-se sempre a mantissa normalizada entre 1 e 2. Desta forma, a sua parte inteira é sempre um bit igual a 1 (um) que neste caso, é necessária sua representação.

A representação do ponto flutuante proposto para 32 bits, é descrita como:

$$(-1)^S \; x \; M(2)^E \tag{3}$$

As principais diferenças entre o formato proposto e o IEEE 754 são que no formato proposto o bit 1 mais significativo da mantissa é mantido na representação (neste caso a mantissa é dita normalizada), e que na representação proposta o expoente é representado em complemento de dois, podendo possuir valores negativos e positivos, enquanto que no formato IEEE 754 o expoente assume apenas valores positivos.

Com isso, pode ser observado na Tabela 2, a representação, tanto no padrão IEEE, quando para o formato de representação em ponto flutuante proposta, respectivamente, para o número 1.2, separados em bit do sinal, expoente e mantissa.

Tabela 1 - Representação Padrão IEEE 754: Sinal, Expoente e Mantissa.

	IEEE 754	Proposto
Sinal	0	0
Expoente	0111111	11101010
Mantissa	001100110011 00110011010	100110011001 10011001101

De acordo com a Tabela 2, nota-se que para o formato proposto, tem-se a mantissa normalizada, diferentemente do padrão IEEE 754. O expoente também é diferente devido ao fato da representação em complemento de 2.

Para que as operações possam ser realizadas de acordo com a representação em ponto flutuante proposta, existe uma separação dos números a serem operados, em mantissa, bit do sinal e expoente. Por exemplo, quando se deseja multiplicar dois números, sendo eles genéricos, N1 e N2, e o resultado dessa operação N, a representação no formato de ponto flutuante proposta para essa operação é:

$$N_1 = (-1)^{S_1} \times \; M_1(2)^{E_1} \tag{4}$$

$$N_2 = (-1)^{S_2} \times \; M_2(2)^{E_2} \tag{5}$$

$$N = N_1 \; x \; N_2 = (-1)^{(S_1 \wedge S_2)} \times \; (M_1 \; x \; M_2) \times \; (2)^{(E_1 + E_2)} \tag{6}$$

Para que a operação de multiplicação seja realizada, faz-se a multiplicação das mantissas, e somam-se os expoentes, e para o bit do sinal, a multiplicação de dois números com mesmo sinal, é sempre positivo, sendo dado por uma lógica XOR.

Para operação de divisão, é semelhante, utilizando os mesmos números genéricos, há primeiro a separação dos bits, em mantissa, bit do sinal e expoente. Logo, é feita a subtração dos expoentes e as mantissas são divididas. O bit do sinal é definido da mesma forma que a multiplicação, sendo assim:

$$N = N_1 \times N_2 = (-1)^{(S_1 \hat{} S_2)} \times (M_1 / M_2) \times (2)^{(E_1 - E_2)} \quad (7)$$

Quando deseja-se somar dois números nesta representação, como por exemplo os números já exemplificados anteriormente, primeiro deve-se desnormaliza-los, que é simplesmente igualar os expoentes, em que o expoente maior é mantido, e o outro igualado a ele. Feito isso, a representação dos números para o formato proposto se dá como:

$$N_{1D} = (-1)^{S_1} \times M_{1D}(2)^{E_D} \quad (8)$$

$$N_{2D} = (-1)^{S_2} \times M_{2D}(2)^{E_D} \quad (9)$$

Após a desnormalização desses números, é necessário obter a mantissa sinalizada (SM), que é obtida de acordo com o valor do bit do sinal de cada número. Se o bit do sinal for 0, a mantissa sinalizada possui o mesmo valor da mantissa e caso o bit do sinal seja 1, a mantissa sinalizada é o complemento de dois da mantissa.

$$N_{1D} = SM_{1D}(2)^{E_D} \quad (10)$$

$$N_{2D} = SM_{2D}(2)^{E_D} \quad (11)$$

Dessa forma, a operação de soma/subtração pode ser efetuada, ressaltando que o sinal da operação é obtido de forma simples; quando o bit mais significativo de uma das mantissas sinalizadas for "1", é feito o complemento de 2 desse número, e assim a soma das mantissas sinalizadas é realizada, caracterizando a operação de subtração, caso contrário, somam-se as mantissas normalmente, e o resultado da operação é dado por:

$$N = N_{1D} \times N_{2D} = SM_{1D} + SM_{2D} \times (2)^{E_D} \quad (12)$$

Logo, para que o número siga o formato da representação em ponto flutuante proposto, deve ser normalizado após a operação, em que o bit mais significativo da mantissa deve ser igual a "1", como apresentado no início desta seção.

Os circuitos utilizados para cada uma das operações exemplificadas anteriormente, são descritos nas próximas seções.

3.2 Circuitos Aritméticos

A seguir, serão descritos os circuitos aritméticos que realizam as operações utilizando o formato de ponto flutuante proposto.

3.2.1 Multiplicação

Na operação de multiplicação, os dois números recebidos, já em formato ponto flutuante, há uma separação dos seus bits de sinal, os expoentes de cada número, assim como as mantissas deles; e para a saída da operação a mesma separação, da mesma forma, separados em bit do sinal, expoente e mantissa da saída da operação.

Dessa maneira, para que a operação possa ser efetuada, faz-se a multiplicação das mantissas, e somam-se os expoentes. O circuito utilizado para a operação de multiplicação, pode ser analisado de acordo com a Figura 1.

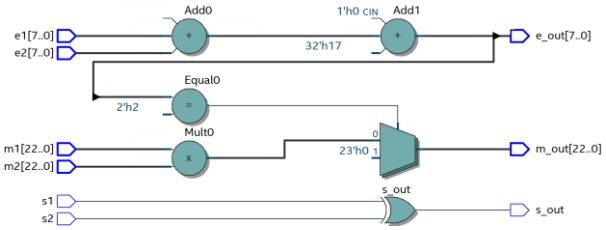

Figura 1. Circuito de Multiplicação

Este circuito, é composto por um multiplicador, que vai realizar a multiplicação das mantissas, mantendo os bits mais significativos do resultado, já que quando se multiplica dois números o resultado é maior em número de bits, garantindo assim os 23 bits de mantissa.

Ainda, dois somadores em ponto fixo são utilizados neste circuito de multiplicação, para realizar a soma dos expoentes; um comparador também é utilizado, para verificar se os expoentes são iguais. Para o bit do sinal, uma porta lógica X-OR é utilizada.

Logo, quando os sinais dos números a serem multiplicados forem positivos (zero), a saída da porta é zero, mantendo o número positivo, o mesmo ocorre quando o sinal dos números for negativo (um), a saída da porta lógica será zero, devido a multiplicação de dois números com sinais negativos ter como resultado um número positivo. Então, quando os números apresentarem sinais diferentes, ou seja, um positivo e o outro negativo, a saída da porta lógica apresentada será um, fazendo com o que o sinal do resultado da multiplicação seja negativo.

3.2.2 Divisão

Para que a divisão entre dois números seja realizada, acontece uma separação do bit do sinal, do expoente e da mantissa, semelhante ao descrito na multiplicação. Porém, na realização da operação, há a divisão das mantissas, e também a subtração dos expoentes.

O circuito usado para a operação de divisão, pode ser visto de acordo com a Figura 2, para que assim possa ser analisado.

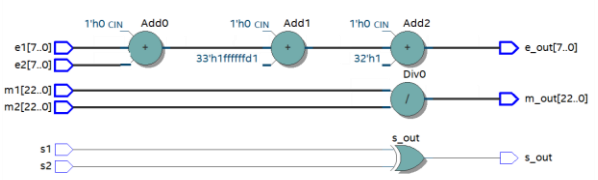

Figura 2. Circuito de Divisão

Em que, este circuito é composto por um divisor, que faz a separação das mantissas, e logo pegando os bits mais significativos do resultado desta divisão, que para o caso estudado de 32 bits, são 23 para a mantissa.

Ainda este circuito da operação de divisão, é composto por três somadores, para que ocorra a subtração dos expoentes. Para o bit do sinal, ocorre similar ao circuito da multiplicação, em que uma porta X-OR é usada.

3.2.3 Soma/Subtração

Para que dois números possam ser somados, de acordo com essa notação científica de representação de ponto flutuante, tem-se que primeiro igualar os expoentes dos números a serem somados. Existe um circuito que faz a "desnormalização" dos números, ou seja, igualar os expoentes, mostrado na Figura 3.

O circuito da Figura 3, é responsável por fazer a desnormalização do número, ou seja, com os expoentes igualados, em que os dois números são separados em mantissa, expoente e bit do sinal. E para igualar os expoentes, são utilizados somadores e multiplexadores, sua saída é apenas um expoente, já que os dois números agora, possuem o mesmo expoente, para que o processo da soma seja realizado

Feito a desnormalização, repete-se o expoente, e as mantissas são somadas. Em que o circuito a da Figura 4, representa o processo de soma.

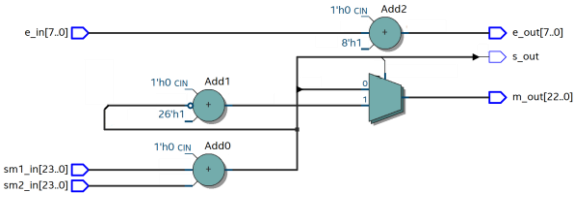

Figura 4. Circuito de Soma

Neste circuito, é utilizado um multiplexador, que é um dispositivo que seleciona as informações, neste caso, de duas fontes de dados em um único canal, ainda são usados três somadores de ponto fixo, em que dois deles são responsáveis por realizar a soma das mantissas; como os expoentes foram igualados, então, eles se repetem para o resultado da operação. O bit do sinal, é simples, pois fazendo a soma de dois números positivos, o resultado é positivo, logo, a soma de dois negativos, tem-se como resultado um número negativo.

Então, como o formato proposto a mantissa tem que estar normalizada, com 1 no bit mais significativo dela, é necessária à sua normalização novamente. O circuito responsável pela normalização do número, é composto por vários multiplexadores, e um somador, para que assim a mantissa possa ser normalizada.

A operação de subtração, pode ser realizada fazendo a soma de um número positivo com um número negativo.

4 Resultados

Além de operações de soma/subtração, multiplicação e divisão, que serão abordadas de forma a comparar com o padrão IEEE, que é o mais utilizado na atualidade, ainda será apresentado tabelas de lógicas que são utilizadas para a realização das operações, assim como a implementação de um filtro FIR a fim de verificar o bom funcionamento desta representação em ponto flutuante proposta.

Para a realização destes testes em simulação, tanto na representação em ponto flutuante proposta, quanto no padrão IEEE, é usado o chip da família Cyclone IV E da fabricante Altera (CYCLONE IV, 2011).

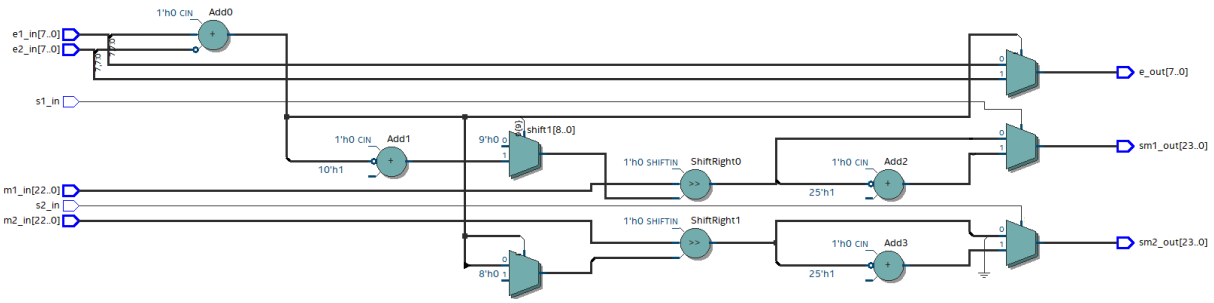

Figura 3. Circuito de Desnormalização.

4.1 Operações: Formato Proposto X IEEE

Através da representação em ponto flutuante proposto, foram realizadas operações aritméticas, de soma/subtração, multiplicação e divisão, a fim de comparar os resultados com as operações através do padrão IEEE 754, para assim verificar a eficácia do método proposto.

É importante destacar que tanto o padrão IEEE, quanto a representação que é proposta, possuem extremos de representações numéricas, e que as operações apresentadas na Tabela 3 a seguir, são operações que vão de números medianos até os extremos.

Tabela 2. Operações

INA	INB	OP	IEEE	Proposto
2,5	3,14	/	0,796178	0,796178
2,5	3,14	-	-0,640000	-0,6400003
-192,27	15,38	×	-2957,112	-2957,112
1,70E+38	1,70E+38	+	3,40E+38	2,94E-39
1,70E+38	2	×	3,40E+38	0
3,55E-06	9,76E-08	-	3,45E-06	3,45E-06
3,55E-06	9,76E-08	×	3,46E-13	3,46E-13
1,70E+38	0,00E+00	/	Infinito	0,00E+00
1,70E+38	0,00E+00	+	1,70E+38	1,70E+38
1,00E+37	-2,00E-20	/	NaN	0,00E+00
1,00E+37	-2,00E-20	×	-2,00E+16	0,00E+00

De posse desses resultados, fica passível realizar uma comparação entre os métodos de representação em ponto flutuante, em que para operações realizadas com números entres os extremos, pode ser observado que a representação proposta possui o mesmo resultado que o padrão IEEE.

Quando são realizadas operações próximas aos extremos, ou quando é realizado uma operação de divisão ou multiplicação de números muito grandes por números muito pequenos em módulo, para o padrão IEEE aparecem os caracteres especiais (seção 2.5.2), como NaN, e infinito, e os resultados obtidos através do formato de representação proposto apresentam divergência de resultados.

Divergência esta, que é explicada de forma que para a representação em ponto flutuante proposta não possui esses caracteres especiais, devido ao fato de que, dependendo da aplicação os mesmos podem aumentar o tempo de resposta das operações, e que na maioria das aplicações não é necessário.

4.2 Elementos Lógicos: Formato Proposto X IEEE

Os recursos utilizados por cada circuito aritmético em ambos os formatos de representação estão mostrados na Tabela 4. Em que LE representa os elementos lógicos do FPGA, REG os registradores, e MULT os multiplicadores utilizados.

Tabela 3. Elementos Lógicos das Operações

OP	IEEE			Proposto		
	LE	REG	MULT	LE	REG	MULT
+	258	119	144	635	0	0
×	227	267	7	83	0	7
/	263	316	16	1675	0	7
-	258	119	144	635	0	0

Observando a Tabela 5, percebe-se que o formato de representação em ponto flutuante proposto, apresenta uma grande vantagem em relação ao padrão IEEE quando se diz respeito a lógicas utilizadas. Uma das principais vantagens, se não a principal, se dá em que o formato proposto não utiliza de ciclos de clock (registradores) como o padrão IEEE, suas operações utilizam apenas de circuitos combinacionais para serem efetuadas.

Em relação as lógicas utilizadas, a representação em ponto flutuante proposta, não utiliza nenhum tipo de registrador para efetuar qualquer que seja a operação, enquanto o padrão IEEE, utiliza na casa de centenas de registradores. Somente nas operações de multiplicação e divisão, são usados multiplicadores no formato proposto, enquanto no padrão IEEE eles são utilizados também na soma/subtração, e em grande quantidade. Quanto aos elementos lógicos, são muito mais que o dobro quando comparados com o formato proposto.

O número alto de LE nas operações de soma e subtração se dá pelo fato de que ambas necessitam dos circuitos de normalização e denormalização da mantissa. O circuito de soma/subtração propriamente dito gasta 79 LE, enquanto que o de normalização e denormalização gasta 155 LE e 401 LE respectivamente.

4.3 Filtro FIR: Formato Proposto X IEEE

Ainda a título de comparação dos métodos, representação em ponto flutuante proposta e padrão IEEE 754, é implementado um filtro FIR, para que um sinal de entrada possa ser comparado em sua saída.

Um filtro FIR (*Finite Impulse Response*), possui este nome por possuir resposta ao impulso finita. O que diz que ao excitar um filtro FIR de ordem N com um impulso, a resposta do filtro cairá a zero, após N+1 amostras do sinal de entrada.

Este filtro é um sistema linear invariante no tempo, portanto, sua resposta pode ser calculada utilizando a convolução, que para um filtro de ordem N, cujos coeficientes são h[k], tem-se a saída do filtro FIR dada por:

$$y[n] = \sum_{k=0}^{N} h[k]x[n-k] \qquad (13)$$

O diagrama de blocos que representa a implementação da equação a diferenças do filtro é apresentado na Figura 5:

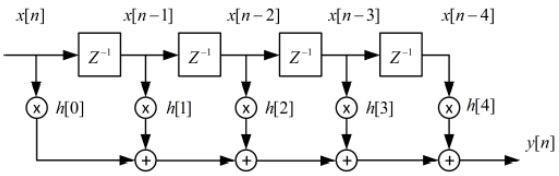

Figura 5. Diagrama de Blocos Filtro FIR

Os filtros FIR apresentam algumas propriedades úteis que podem os tornar preferíveis entre outros filtros, como por exemplo: são inerentemente estáveis; não usam realimentação, em consequência os erros de arredondamento não se acumulam; podem ter fase linear; e ainda podem ter fase mínima (Vahid, 2009)

4.3.1 Resultado: Filtro FIR

Através das operações realizadas com a representação em ponto flutuante proposto, o filtro FIR é implementado, de acordo com as operações descritas no Capítulo 3 no diagrama de blocos do filtro mostrado na seção anterior, para que sua saída seja comparada com a saída de um outro filtro FIR com o padrão IEEE, a fim de verificar a eficácia do método proposto.

Dessa forma, pode ser observado na Figura 6, que se segue o sinal de entrada para o filtro, assim como os sinais de saída descritos. Em que, quando se compara o sinal do método proposto com o padrão IEEE com 32 bits, nota-se que os sinais são idênticos, sendo assim representada a exatidão e otimalidade da representação em ponto flutuante proposta quando comparada ao padrão IEEE, em uma forma mais complexa, sendo através de um filtro.

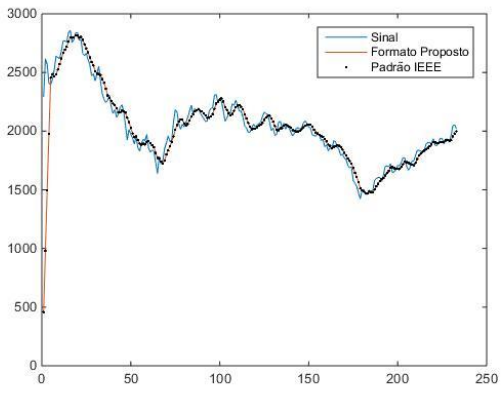

Figura 6. Comparação das Saídas Filtro FIR

Foi visto que para 32 bits, o sinal de saída do filtro para a representação em ponto flutuante proposto, pode ser considerada idêntica ao padrão IEEE, para 1 bit de sinal, 8 bits de expoente e 23 bits de mantissa. Para fins de estudo, é interessante diminuir o número de bits, tanto do expoente quanto da mantissa para o formato proposto, a fim de avaliar a necessidade da utilização dos 32 bits para esta aplicação.

Dessa forma, inicialmente diminuindo bit a bit na mantissa, e fazendo as comparações com o sinal do filtro e a saída IEEE, nota-se que de acordo com o decréscimo dos bits da mantissa, a saída do filtro FIR por meio do formato proposto vai apresentando um erro em relação a saída IEEE, até que ao chegar em 10 bits de mantissa, o erro apresentado é muito grande, devido ao fato de que para dez bits de mantissa, o número não consegue ser representado, sendo inviável a comparação com sinal original.

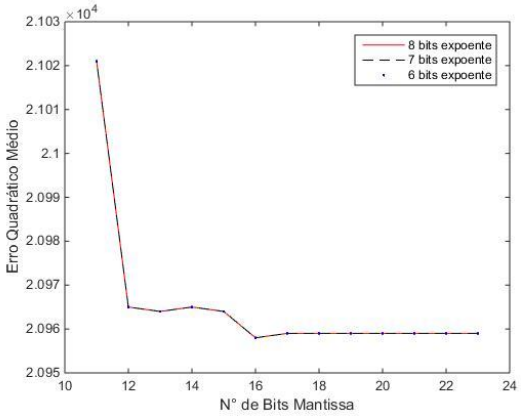

Figura 7. Erro do filtro FIR, com a Alteração de Bits na Mantissa e Expoente Para o Formato Proposto

De maneira análoga, porém diminuindo bit a bit no expoente, o sinal de saída do formato proposto, apresenta erros gradativos à medida que se decresce os bits, logo quando se têm 5 bits de expoente, o sinal de saída proposto já não é possível fazer comparações com a saída IEEE.

Fazendo-se então, um decréscimo bit a bit da mantissa quando se diminui em um bit o expoente, são apresentados erros, que de acordo com o decréscimo dos bits, o erro aumenta. Com isso, tem-se como resultado que o erro apresentado é o mesmo para quando diminui os bits da mantissa bit a bit para expoentes com 8, 7, e 6 bits.

Com isso, para a implementação do Filtro FIR, com o menor números de bits que possa ser representado, ou seja, com 11bits de mantissa e 6 de expoente, além do bit de sinal. Isso reduz o número de elementos lógicos que devem ser utilizados, tornando a implementação do filtro mais eficiente.

5 Conclusão

A representação em ponto flutuante proposta neste trabalho, apresentada desde seu conceito, sua representação e resultados através de simulações, obteve resultados satisfatórios. Já que, quando comparada com o padrão IEEE, na maioria das vezes obteve resultados equivalentes.

É importante destacar que nesta proposta, quando se diz respeito às operações de soma/subtração, divisão e multiplicação, os resultados são muito bons, ressaltando que para valores próximos aos extremos, e indeterminações como 0/0, não são representados pelo método proposto, devido ao número de elementos lógicos que seriam adicionados e ao fato de nem toda aplicação necessitar destes caracteres especiais.

Em relação aos elementos lógicos na realização das operações, como já destacado durante o trabalho, são menos da metade quando comparados às operações realizadas pelo IEEE, sendo este um resultado de destaque para a pesquisa em questão.

Com a implementação do filtro FIR, pelo fato de ser um circuito mais complexo, esperava-se que houvesse alguns erros. Porém, o resultado obtido quando comparado ao padrão IEEE, é satisfatório, ressaltando que o tempo de operação ser mais rápido devido ao fato de apenas utilizar circuitos combinacionais, e não ciclos de clock como no padrão IEEE.

Quando foi realizado a simulação para o filtro com o decréscimo de bits da mantissa e expoente, o resultado foi satisfatório, por apresentar erros satisfatórios em relação à alteração do número de bits da mantissa por decaimento de um bit de expoente. Além disso, o número de elementos lógicos para cada operação em relação à modificação de cada bit foi bem sucedida, já que o número de lógicas diminui com o decréscimo de bits.

Portanto, o formato proposto de representação do ponto flutuante, apesar de poder melhorar em aspectos representativos de caracteres especiais, como mencionado, apresenta ótimos resultados, tanto nas operações simples, quanto para circuitos mais complexos, como o filtro FIR, da mesma forma que mostrou ótimos dados quando realizado teste com diferentes números de bits na sua representação.

Sendo assim, os próximos passos de pesquisa se dedicam a implementar o formato proposto de representação do ponto flutuante em filtros e circuitos mais complexos, a fim de obter uma resposta mais rápida e com maiores índices de exatidão nos resultados.

Agradecimentos

Os autores deste trabalho agradecem a Universidade Federal de Juiz de Fora, a CAPES, ao CNPQ e a FAPEMIG pelo suporte a essa pesquisa.

Referências Bibliográficas

Singh, Prateek, and Kalyani Bhole. "Optimized floating point arithmetic unit." India Conference (INDICON), 2014 Annual IEEE. IEEE, 2014.

Dou, Y., Vassiliadis, S., Kuzmanov, G. K., & Gaydadjiev, G. N. (2005, February). 64-bit floating-point FPGA matrix multiplication. In Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays(pp. 86-95). ACM.

Hemmert, K. S., & Underwood, K. D. (2005, April). An analysis of the double-precision floating-point FFT on FPGAs. In Field-Programmable Custom Computing Machines, 2005. FCCM 2005. 13th Annual IEEE Symposium on (pp. 171-180). IEEE.

Govindu, G., Choi, S., Prasanna, V., Daga, V., Gangadharpalli, S., & Sridhar, V. (2004, April). A high-performance and energy-efficient architecture for floating-point based LU decomposition on FPGAs. In Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International(p. 149). IEEE.

DeLorimier, M., & DeHon, A. (2005, February). Floating-point sparse matrix-vector multiply for FPGAs. In Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays (pp. 75-85). ACM.

David, A. P., & John, L. H. (2005). Computer organization and design: the hardware/software interface. San mateo, CA: M organ Kaufmann Publishers, 1, 998.

CYCLONE IV, D. H. (2011). Altera Corporation, San Jose, CA, USA (Vol. 3). CYIV-53001-1.5.

Vahid, F. (2009). Sistemas Digitais. Bookman Editora.