Arquitetura Paralela Reconfigurável em FPGA para Implementação de Operadores Elementares da Aritmética Intervalar

D. N. Anselmo
C. A. P. S. Martins (Orientador)
G. L. Soares (Orientador)
Pontifícia Universidade Católica de Minas Gerais
Programa de Pós Graduação em Engenharia Elétrica
Belo Horizonte, Minas Gerais – Brasil - 30535610
dayse@ieee.org

Centro Universitário UNA
Belo Horizonte, Minas Gerais – Brasil –30494310
UI - Universidade de Itaúna
Trevo Itaúna / Pará de Minas, Itaúna – Minas Gerais,
35680-142, Brasil
dayse@ieee.org

***Resumo* — Diversas áreas do conhecimento necessitam de meios para tornar os sistemas de computação mais eficientes em termos de diminuição de tempo de resposta. De modo particular, os programas que empregam rotinas baseadas em aritmética intervalar carecem de redução do tempo no processamento dos cálculos, pois as operações matemáticas nestes algoritmos levam em conta os limites dos intervalos de cada operando, onerando o custo computacional. Diferentes aplicações deixam de usar algoritmos intervalares porque o compromisso do sistema com o tempo de execução é fator determinante ao problema. Desta forma, para favorecer o desempenho de aplicações que utilizam a aritmética intervalar, este trabalho apresenta a utilização de um dispositivo eletrônico com arquitetura reconfigurável para paralelizar o cômputo de operações aritméticas elementares. Foram analisados o uso do paralelismo em conjunto com a computação reconfigurável, com o objetivo de acelerar algoritmos da aritmética intervalar. Os diferentes ganhos de desempenho e os respectivos consumos de recursos produzidos pela flexibilidade produzidos pelas características de reconfiguração da arquitetura proposta comprovam que o uso de um *Field Programmable Gate Array* seria uma possível solução, visto que o mesmo associa a flexibilidade do *software* com o desempenho *hardware*.**

Palavras chaves— aritmética intervalar, arquitetura reconfigurável, FPGA.

I. INTRODUÇÃO

O PITAC (*President's Information Technology Advisory Committee*) [1] reporta que diversos problemas do mundo real difíceis ou complexos para a análise ou solução humana, devido ao volume da massa de dados retirados dos experimentos necessários ao estudo do fenômeno, podem ser tratados com o apoio dos sistemas de computação. Deste modo, a utilização dos recursos dos sistemas de computação para entendimento e solução de problemas complexos, tornou-se assunto de muita importância para a liderança científica, competitividade econômica e segurança nacional [2].

Aplicações como otimização, processamentos de sinais e imagens [3], podem ter desempenho comprometido devido ao excesso de operações aritméticas. Em função disto, a busca por mecanismos que executem cálculos numéricos de forma mais eficiente, é um dos campos de pesquisa fundamentado nos mais variados campos da ciência [2]. Estas aplicações, citadas anteriormente, estão submetidas a diferentes tipos de incerteza. Métodos comumente indicados para lidar com os distúrbios propagados pelas incertezas são análise intervalar[4] [7] e lógica fuzzy [5] [6].

A análise intervalar teve maior divulgação no ramo dos sistemas de computação na década de 1950 e a maior contribuição foi a de Moore [9] em 1962. Trata-se de um método utilizado para monitorar e controlar a propagação de erros em cálculos numéricos. Contudo programas desenvolvidos para implementarem operações da área da análise intervalar, seguem um paradigma de programação sequencial e geram naturalmente um aumento no tempo de execução destas operações, em relação às operações convencionais. Isto se justifica no fato de que as operações intervalares empregam cálculos escalares que envolvem os limites dos intervalos, acarretando um acréscimo no número de operações e com isto na complexidade computacional [8] [9] [10]. Percebe-se então, que este volume de dados tem um custo computacional claramente maior que as operações convencionais, necessitando então, de melhoria no desempenho em termos de tempo de resposta. Mas apesar da segurança e qualidade do resultado das aritméticas intervalares, ainda existem problemas [11] [12].

Conforme apresentado em [13], pesquisadores vêm empenhando-se em estudos sobre a aplicação da aritmética intervalar, bem como a computação precisa da mesma [14] [15] [16]. Esta vem se destacando no cenário da ciência computacional, visto que a mesma torna possível o controle rigoroso sobre a propagação dos erros de precisão e de exatidão dos resultados, bem como nos possibilita o tratamento e modelagem da incerteza em computação, conforme comentado em [17] [18] [19] [20].

Os pacotes de software existentes para a aritmética intervalar são demasiadamente lentos para cálculos numéricos intensos [2] porque apesar de terem sido criados para atender muitas funcionalidades, os computadores de uso geral, tem sua

arquitetura voltada para instruções fixas e com esta restrição eles conseguem atender apenas às aplicações desenvolvidas para este *hardware* em específico. Pensando neste contexto, em 1994 [11] foram publicados assuntos relacionados à importância e aplicabilidade da aritmética intervalar paralela, com destaque para artigos que abordavam a necessidade da paralelização de métodos intervalares, reduzindo seu tempo de execução. Começaram-se então, estudos sobre o uso da computação paralela, como uma possível solução para a melhoria dos *softwares* implementados para computadores de uso geral, possibilitando a execução de partes pequenas do código de forma paralela.

Embora o paralelismo apresente vantagens, infelizmente não são todas as aplicações que podem ter seu desempenho melhorado utilizando a computação paralela e isto pode ser observado ao medirmos o *speedup*. Outras aplicações podem exigir um processamento maior, anulando assim os benefícios da computação paralela [21]. Um *software* apresenta a flexibilidade para realizarmos um número elevado de tarefas diferenciadas, quando da necessidade de alterações em aplicativos, uma vez que o algoritmo pode facilmente ser modificado a qualquer momento e depois incorporado ao código. Entretanto, a complexidade deste *software* cresce em função da demanda da carga de processamento, causando um comprometimento ao *software* em termos de tempo de execução. Uma possibilidade seria o uso de um ASIC (*Application Specific Integrated Circuit*) [22].

O ASIC é um *hardware* que oferece diversos recursos otimizados para a execução rápida de tarefas específicas consideradas críticas. Porém a fabricação destes dispositivos demanda esforço, custo financeiro, e um projeto que pode levar meses ou anos para ser desenvolvido [23]. Uma outra possibilidade então, seria o uso de um FPGA *(Field Programmable Gate Array)*.

O FPGA é um *hardware* integrado cuja aplicação é de uso geral tal como os microprocessadores [24]. Possui um conjunto de blocos lógicos (CLBs) e interligações que podem ser configurados para realizar uma tarefa específica, mas pode ser reconfigurado sob demanda para realizar outras tarefas específicas [25]. Os CLBs contém um par de *Slices* (menor unidade lógica configurável de um FPGA), que são **c**ircuitos idênticos construídos pela reunião de flip-flops chamados de LUTs (*Look-up Tables*). O FPGA possui também os IOBs (*Input/Output Block*), que são circuitos responsáveis pelo interfaceamento das saídas provenientes das saídas das combinações de CLBs. Ao utilizarmos o FPGA, estaremos certamente utilizando um *hardware* que associa a flexibilidade do *software* com o desempenho do *hardware* [26]. Segundo [25], resultados estimativos mostram que a combinação de um computador de propósito geral e dos recursos de um FPGA pode incrementar significativamente o desempenho de processamento de grandes massas de dados. Em [11] é citado que a diminuição no tempo de processamento da aritmética intervalar pode ser produzida com a implementação da mesma tanto via *software* ou via *hardware*, quanto através da combinação dos dois: *software* + *hardware*. Parte das críticas relacionadas ao tempo, não terão mais validade, considerando os benefícios produzidos pelo uso dos recursos de paralelismo e reconfigurabilidade de um FPGA.

Este artigo tem como objetivo implementar as operações de soma, subtração, multiplicação e divisão, da aritmética intervalar dentro de um FPGA, uma vez que a aritmética intervalar tem importância em grande parte dos algoritmos intervalares. A arquitetura paralela reconfigurável em FPGA proposta implementará os operadores intervalares, podendo ser usada em qualquer aplicação que envolva operações intervalares. A otimização se dará com a aplicação de paralelismo em vários níveis, dentro das operações elementares (intra operações) e/ou inter operações, quando da utilização de diversos módulos de operações elementares na execução de outras aplicações que utilizem a composição das mesmas.

O trabalho está organizado da seguinte forma: Seção II apresenta a fundamentação teórica desta pesquisa; Seção III são apresentados os trabalhos relacionados que justificam o desenvolvimento desta pesquisa; Seção IV é apresentada a arquitetura proposta para a solução do problema; Seção V valida a abordagem através de testes e resultados; Seção VI apresenta as conclusões.

II. FUNDAMENTAÇÃO TEÓRICA

A. *Análise Intervalar*

Trindade [3] aponta que os computadores apresentam dificuldades quando da necessidade de representação dos números reais, como por exemplo, em cálculos numéricos onde a precisão e exatidão dos resultados são importantíssimas. Técnicas intervalares podem representar números reais de forma confiável e automatizar o controle de erros causados pela propagação dos mesmos nos dados e nas medidas, tais como falhas em arredondamento e truncamento nas operações aritméticas, por exemplo [18]. A análise intervalar é baseada em conceitos fundamentais para o desenvolvimento de técnicas/algoritmos intervalares [18] [30]. Parte da análise é composta pelas definições apresentadas a seguir.

Definição 1: Um intervalo fechado $[z]$ de números reais é definido por:

$$[\underline{z}, \overline{z}] = \{z \in \mathbb{IR} \mid \underline{z} \leq z \leq \overline{z}\} \quad (1)$$

sendo $\underline{z}$ e $\overline{z}$ representam, respectivamente os limites inferior e superior de $[z]$. O domínio de todos os intervalos de números reais é dado por $\mathbb{IR}$.

Definição 2: Denote um operador binário $\Diamond$ para representar as quatro operações elementares da aritmética intervalar +, -, * e /.

$$[z] \Diamond [w] = \{z \Diamond w \mid z \in [z], w \in [w]\} \quad (2)$$

então,

a) a soma $[z]$ *e* $[w]$ *é:*

$$[\underline{z}, \overline{z}] + [\underline{w}, \overline{w}] = [(\underline{z} + \underline{w}); (\overline{z} + \overline{w})] \quad (3)$$

b) a subtração $[z]$ *e* $[w]$ *é:*

$$[\underline{z}, \overline{z}] - [\underline{w}, \overline{w}] = [(\underline{z} - \overline{w}); (\overline{z} - \underline{w})] \quad (4)$$

c) a multiplicação $[z]$ *e* $[w]$ *é:*

$$[\underline{z}, \overline{z}] * [\underline{w}, \overline{w}] = \begin{bmatrix} min(\underline{z} * \underline{w}, \underline{z} * \overline{w}, \overline{z} * \underline{w}, \overline{z} * \overline{w}); \\ máx(\underline{z} * \underline{w}, \underline{z} * \overline{w}, \overline{z} * \underline{w}, \overline{z} * \overline{w}) \end{bmatrix} \quad (5)$$

d) divisão $[z]$ *e* $[w]$ *é:*

$$\frac{[\underline{z},\overline{z}]}{[\underline{w},\overline{w}]} = \left[min\left(\frac{\underline{z}}{\overline{w}},\frac{\underline{z}}{\underline{w}},\frac{\overline{z}}{\overline{w}},\frac{\overline{z}}{\underline{w}}\right); máx\left(\frac{\underline{z}}{\overline{w}},\frac{\underline{z}}{\underline{w}},\frac{\overline{z}}{\overline{w}},\frac{\overline{z}}{\underline{w}}\right) \right] \quad (6)$$

com $0 \notin w$.

Pode-se observar claramente que as operações da aritmética intervalar consideram os limites dos intervalos de cada operando, onerando o custo computacional.

III. TRABALHOS RELACIONADOS

Em seus trabalhos, [9] [18] [29] e outros citam que aplicações que utilizam a aritmética intervalar, executam um excesso de operações aritméticas. Tais operações geram um aumento no tempo de execução resultando em custos no processamento, tornando inviável o uso da aritmética intervalar em tais aplicações. [18] e [40] reforçam que a computação da aritmética intervalar necessita de melhorias no desempenho em termos de tempo de resposta. Mas em [39] é mostrado que os pacotes de *software* existentes para a aritmética intervalar requerem redução do tempo no processamento dos cálculos.

Pesquisadores [41] e [42] afirmam que um projeto implementado em *hardware* poderia proporcionar uma melhoria significativa de desempenho sobre implementações de *software* que executam estas mesmas operações intervalares. [43] e [44] descrevem sobre os ganhos do uso da representação binária juntamente com a representação intervalar e mostram que o poder combinado de uma linguagem simbólica de programação lógica, com uma matemática logicamente eficiente, é mais poderoso do que isoladamente considerados.

Em [28] é comentado que em quase todas as áreas do conhecimento, principalmente as relacionadas às ciências exatas e engenharias, tem se observado um crescente aumento no uso da computação na solução de problemas complexos. Para muitos destes problemas, as soluções precisam ser obtidas em intervalos de tempo cada vez menores, ou até em tempo real e que para grande parte destes problemas complexos, as soluções implementadas em *software* sequencial executado em um processador de propósito geral não atendem às necessidades de tempo de resposta e/ou desempenho, de um modo geral.

Em [24] argumenta-se que diferentemente dos ASICs, os FPGAs não são programados durante o processo de fabricação e distribuição, porque podem ser programados e reprogramados muitas vezes, inclusive dinamicamente. Ele ainda diz que com os FPGAs obtêm-se uma solução que agrega a vantagem do desempenho do *hardware* e da flexibilidade do *software*. Estes dispositivos de *hardware* podem implementar cálculos e simultaneamente (paralelismo) computar diversas operações, em recursos distribuídos em um chip de silício, em tempo computacional centenas de vezes mais rápido que em microprocessadores.

Com base no anteriormente mostrado, em [14] é proposta uma nova abordagem para mostrar uma forma eficaz de adição intervalar, subtração intervalar e multiplicação intervalar, utilizando ponto flutuante. Ao final concluiu-se que o somador/subtrator proposto teve o mesmo desempenho dos somadores/subtratores convencionais, porém com custo menor. O multiplicador apresentou a metade do desempenho de um multiplicador de ponto flutuante convencional. Em [29] é apresentado um projeto em *hardware*, de um sistema seletor e comparador de intervalos, executado em um FPGA. Ao final verificou-se uma grande aceleração das operações intervalares em FPGA comparado a um *software* que executa o mesmo procedimento.

Sendo assim, novas soluções devem e estão sendo desenvolvidas e utilizadas com o objetivo de melhorar o tempo de resposta. Dentre estas soluções, podemos citar a computação e arquiteturas reconfiguráveis. Algumas soluções de computação de alto desempenho têm sido desenvolvidas em pesquisas realizadas e divulgadas: *Symposium on Computer Architecture and High Performance Computing* (SBAC-PAD), *Workshop* em Sistemas Computacionais de Alto Desempenho (WSCAD) e em congressos da SBC (Sociedade Brasileira de Computação) [28] .

IV. ARQUITETURA PARALELA RECONFIGURÁVEL EM FPGA

A. *Principais Causas do Problema Motivador*

Para a realização das operações escalares intervalares, são necessárias execuções sequenciais das operações escalares convencionais suportadas pelo hardware fixo (não reconfigurável) dos processadores tradicionais. Existem repetições no eixo do tempo de operações que poderiam ser executadas simultaneamente se a arquitetura tivesse o número suficiente de módulos aritméticos fundamentais para executar as operações elementares básicas, para a execução das operações escalares intervalares. Para fazermos a soma escalar intervalar é necessário que se faça somas escalares convencionais, conforme mostrado na equação (3) da seção II. Tradicionalmente, para fazermos a soma escalar intervalar, seriam gastos duas somas sequenciais escalares convencionais, conforme mostrado na Fig.1.

Fig. 1. Configuração Serial da Soma Escalar Intervalar.

Ao aplicarmos paralelismo inter operações, poderíamos fazer as duas operações da Fig.1 simultaneamente e obter um ganho no tempo de processamento das operações, conforme apresentado na Fig.2.

Fig. 2. Configuração Ideal da Soma Escalar Intervalar.

B. *Apresentação Sistêmica da Arquitetura Proposta*

Inicialmente pensou-se em módulos de operadores básicos da aritmética escalar convencional e com a manipulação destes, seriam realizadas as operações escalares intervalares. A seguir apresentamos algumas possibilidades de configuração da arquitetura modular reconfigurável, para uma determinada aplicação:

- Caso a aplicação aceite um tempo pequeno de execução e permita um grande uso de recursos lógicos disponíveis, pode-se utilizar a configuração ideal mostrada na Fig.3, em que o processamento é feito com paralelismo máximo.
- Caso a aplicação aceite um tempo médio de execução e permita um médio uso de recursos lógicos disponíveis, pode-se utilizar a configuração parcialmente ideal, mostrada na Fig.3, em que o processamento é feito com um paralelismo parcial.
- Caso a aplicação aceite um tempo alto de execução e um pequeno consumo de recursos lógicos disponíveis, pode-se utilizar a configuração serial mostrada na Fig. 3, em que o processamento é feito gastando um único módulo básico e o processamento total é feito sequencialmente.

Fig. 3. Apresentação Sistêmica da Arquitetura.

C. *Síntese*

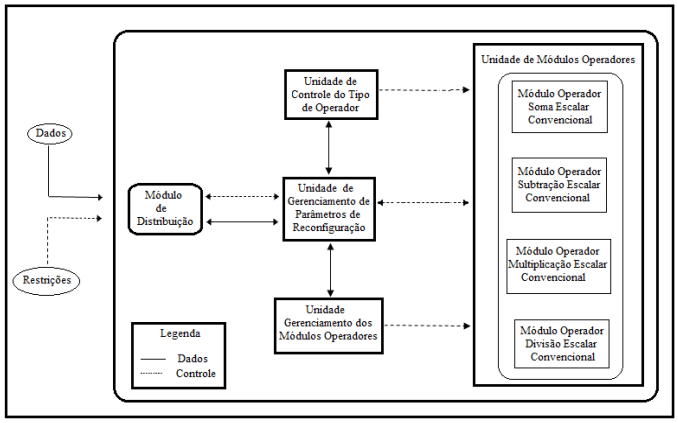
A arquitetura proposta tem como meta atender requisitos de desempenho, flexibilidade e tempo de resposta. Esta possui graus de paralelismo variáveis de acordo com a demanda de cada aplicação. Para tanto, foram utilizados os recursos de paralelismo e reconfiguração, considerando as etapas dos algoritmos das operações elementares da aritmética escalar convencional e escalar intervalar, bem como das operações matriciais convencionais e matriciais intervalares. Esse projeto de arquitetura foi verificado através da implementação em FPGA de uma biblioteca de operações básicas otimizadas, através de módulos codificados em VHDL, de operações de soma, subtração, multiplicação e divisão escalar convencional e escalar intervalar. A escolha da linguagem foi em função da flexibilidade oferecida na descrição do comportamento de circuitos eletrônicos digitais.

Ao selecionar, parametrizar e interligar esses módulos básicos, a arquitetura reconfigurável implementa os operadores intervalares, podendo ser usada em qualquer aplicação que envolva operações intervalares. A otimização se dá com aplicação de paralelismo espacial, paralelismo dentro das operações básicas e paralelismo intra x inter operações, quando da utilização desses módulos básicos na execução de outras aplicações que utilizem a composição das mesmas.

O uso da reconfiguração veio com o objetivo de tornar a arquitetura ainda mais flexível (em termos de requisitos da aplicação), visto que ela permite alterar algumas de suas características, como por exemplo, a quantidade de módulos usados, quantidade de bits de cada operador, tipo de operação realizada. Aumentando a flexibilidade espera-se poder fazer a opção entre ter, por exemplo, um menor tempo de resposta, em detrimento de um maior consumo de recursos lógicos, ou vice-versa. Uma outra vantagem de se usar a computação reconfigurável é que pode-se ter uma diminuição significativa de custos, uma vez que o recurso da reconfigurabilidade permite a utilização do mesmo *hardware* para diferentes versões, quando do uso dos módulos desenvolvidos.

A Fig.4 apresenta o diagrama de blocos da arquitetura proposta, considerando um fluxo hipotético de dados.

Fig. 4. Diagrama de blocos da arquitetura paralela reconfigurável em FPGA proposta, para implementação de operadores elementares da aritmética intervalar.



- **Unidade de Controle do Tipo de Operador:** responsável pela escolha do tipo de operação que foi definida pela aplicação (soma, subtração, multiplicação, divisão).
- **Módulo de Gerenciamento de Parâmetros de reconfiguração:** recebe as informações referentes aos requisitos das aplicações. Com essas informações e baseando-se na quantidade de recursos lógicos disponíveis no *hardware*, este módulo realizará a configuração para o processamento dos demais módulos envolvidos na aplicação.
- **Gerenciamento dos Módulos Operadores:** responsável por controlar os módulos escolhidos na configuração determinada pelo Módulo de Parâmetros de reconfiguração.

Optou-se por simular e validar a arquitetura desenvolvida utilizando um modelo FPGA *Kintex-7* XC7K160TFBG676 da *Xilinx*, 200 MHz; 2048MB + 512MB DDR3 SDRAM, devido aos recursos que o mesmo oferecia para o desenvolvimento desta pesquisa. Deve-se ressaltar que apesar da escolha do FPGA *Kintex-7*, a arquitetura implementada apresenta características de independência do dispositivo utilizado (família, fabricante).

A arquitetura poderá ser modificada durante seu funcionamento (em tempo real), adequando-se melhor à aplicação a ser executada. De acordo com as características citadas, pode-se configurar aplicando o paralelismo total no uso dos módulos básicos, mas também utilizar paralelismo em múltiplos níveis, pois pode-se alterar uma área do *chip* em termos funcionais, enquanto os demais circuitos do mesmo

chip estão em funcionamento, sem prejudicá-los. As operações executadas pela arquitetura desenvolvida serão realizadas apenas por números em ponto fixo.

V. ANÁLISE QUANTITATIVA DOS RESULTADOS

Para comprovar a exatidão numérica dos resultados produzidos pelos módulos, verificou-se que os resultados das operações realizadas pelo FPGA *Kintex-7* XC7K160TFBG676 da *Xilinx,* usando palavras de 16 bits, foram idênticos aos produzidos pelas mesmas operações implementadas em um processador com um *software* codificado em C:

- Dual Core 64 Bits, 1,9 GHz de cada núcleo, 2GB de Ram;
- 64 KB para cache L1 de Instruções, 64 KB de cache L1 de dados, Cache L2, 256 KB.

Os valores que compõem os gráficos das configurações testadas, para análise de tempo dos módulos implementados para o FPGA, foram registrados em simulação pós-síntese.

A. *Resultado no uso das configurações*

É importante ressaltar que o uso das configurações não interfere no valor do resultado final. A grande diferença está no tempo de execução e no consumo de recursos lógicos disponíveis do FPGA. As configurações utilizadas foram apresentadas na seção IV.

Os módulos desenvolvidos para realizar especificamente as operações intervalares, foram chamados de: Módulos Ideais.

Caso aritmético 1A/1B: soma escalar intervalar, utilizando módulos de soma escalar convencional na configuração serial e na configuração ideal.

Caso aritmético 1C: soma escalar intervalar, utilizando o módulo ideal desenvolvido para a realização da soma intervalar.

Caso aritmético 2A/2B: subtração escalar intervalar, utilizando módulos de subtração escalar convencional na configuração serial e na configuração ideal.

Caso aritmético 2C: subtração escalar intervalar, utilizando o módulo ideal desenvolvido para a realização da subtração intervalar.

As Tabelas 1, 2 e 3 mostram os resultados obtidos ao variarmos a aplicação, utilizando as configurações apresentadas na seção IV. Essas tabelas foram construídas com base nos relatórios gerados pela ferramenta de síntese do ambiente de desenvolvimento *Xilinx*.

De acordo com a Tabela 1, na configuração serial o tempo de processamento é maior, como era de se esperar, pois as operações são feitas sequencialmente, mas é possível obter um ganho significativo em termos de recursos disponíveis no FPGA. Na configuração ideal ocorre o paralelismo máximo e há tendência de um gasto maior de recursos lógicos em função de um tempo de execução menor. Vemos também que o módulo ideal implementado especificamente para o processamento da soma escalar intervalar (1C), apresentou ganho superior aos das configurações propostas em relação ao tempo de execução, porém com um maior gasto de recursos disponíveis no FPGA.

A configuração 2B e 2C apresentaram o mesmo tempo de execução. Isto pode ser justificado pelo tipo de organização da arquitetura do módulo. Conforme apresentado na seção IV, as operações intervalares são compostas de sucessivas operações sequenciais convencionais. O módulo ideal intervalar foi desenvolvido utilizando descrição por fluxo de dados + comportamental, o que permitiu uma melhoria na arquitetura do módulo de subtração escalar intervalar. Não há muitas opções de variação das configurações 1 e 2 porque as operações são realizadas pelos módulos básicos.

TABELA I : RESULTADOS DAS CONFIGURAÇÕES

CA	Conf Serial	Conf Ideal	Mód Ideal	Recursos	
				Lut / %	IOB / %
1A	10ns	-	-	25 /0,02	50 / 12,5
1B	-	5ns	-	50 / 0,04	100/ 25
1C	-	-	4ns	52 / 0,05	105 /26
2A	4ns	-	-	16 / 0,01	48 / 12
2B	-	2ns	-	32 / 0,03	96 / 24
2C	-	-	2ns	50 / 0,05	100 / 12

Caso aritmético 3A: multiplicação escalar intervalar, utilizando módulos escalares convencionais na configuração serial.

Caso aritmético 3B: multiplicação escalar intervalar, utilizando módulos escalares convencionais na configuração parcialmente ideal.

Caso aritmético 3C: multiplicação escalar intervalar, utilizando módulos escalares convencionais na configuração ideal.

Caso aritmético 3D: multiplicação escalar intervalar, utilizando o módulo ideal desenvolvido para a realização da multiplicação intervalar.

Caso aritmético 4A: divisão escalar intervalar, utilizando módulos escalares convencionais na configuração serial.

Caso aritmético 4B: divisão escalar intervalar, utilizando módulos escalares convencionais na configuração parcialmente ideal.

Caso aritmético 4C: divisão escalar intervalar, utilizando módulos escalares convencionais na configuração ideal.

Caso aritmético 4D : divisão escalar intervalar, utilizando o módulo ideal desenvolvido para a realização da divisão intervalar.

TABELA II: RESULTADOS DAS CONFIGURAÇÕES

CA	Conf Serial	Parc Ideal	Conf Ideal	Mód Ideal	Recursos	
					Lut / %	IOB / %
3A	90ns	-	-	-	325/ 0,3	256/64
3B	-	50ns	-	-	650/ 0,6	512/128
3C	-	-	26ns		1400/1,4	1024/256
3D	-	-	-	10ns	323/0,3	120/30
4A	266ns	-	-	-	671/0,7	320/80
4B	-	138ns	-	-	1017/1,0	740/185
4C	-	-	64ns	-	1709/1,7	1480/370
4D	-	-	-	63ns	900/0,9	90/23

A Tabela 2 mostra que o comportamento das configurações serial e ideal seguem a mesma tendência das configurações serial e ideal apresentadas na Tabela 1. Contudo, observa-se que os casos das configurações parcialmente ideais 3B e 4B e das configurações ideais 3C e 4C, apresentam gastos de recursos de IOBs superior ao disponível pelo FPGA. Para estes casos, seria mais vantagem usar o módulo ideal, que apresenta um tempo de execução menor, gastando poucos recursos do FPGA.

Apesar de terem sido discutidas apenas as configurações apresentadas na Tabela 2, há flexibilidade de outras configurações.

Para a validação dos casos aritméticos 5 e 6, foi utilizada uma multiplicação matricial [16x8] x [8x16], com palavras de 16 bits (ponto fixo). A escolha da aplicação multiplicação de matrizes e do tamanho da matriz foi em função da quantidade de operações sucessivas envolvidas para atingir o resultado: multiplicações e somas escalares convencionais.

Caso aritmético 5A: multiplicação matricial intervalar, utilizando módulos escalares convencionais na configuração serial.

Caso aritmético 5B: multiplicação matricial intervalar, utilizando módulos escalares convencionais na configuração parcialmente ideal.

Caso aritmético 5C: multiplicação matricial intervalar, utilizando módulos escalares convencionais na configuração ideal.

Caso aritmético 6A: multiplicação matricial intervalar, utilizando módulos ideais (multiplicação, soma e comparação) escalares intervalares na configuração serial

Caso aritmético 6B: multiplicação matricial intervalar, utilizando módulos ideais (multiplicação, soma e comparação) escalares intervalares na configuração parcialmente ideal.

Caso aritmético 6C: multiplicação matricial intervalar, utilizando o módulos ideais (multiplicação, soma e comparação) escalares intervalares na configuração ideal.

Caso aritmético 6D: multiplicação matricial intervalar, utilizando o módulo matricial intervalar ideal desenvolvido para a realização da multiplicação matricial intervalar.

TABELA III: RESULTADOS DAS CONFIGURAÇÕES

CA	Serial	Parc Ideal	Ideal	Mód Ideal	Recursos	
					Lut / %	IOB / %
5A	202240 ns				2600/ 2,6	256/64
5B		38400 ns			5200/ 5,1	512/128
5C			25088 ns		10400/ 10,3	1024/256
5D				16ns	1664	128/32
6A	27648 ns				2986/ 2,9	232/58
6A		13312 ns			5972/ 5,9	464/116
6C			3584 ns		11944/ 11,8	968/232
6D				28 ns	4000106/ 3944,9	106/26,5

A Tabela 3 mostra que o comportamento das configurações serial e ideal, seguem a mesma tendência, das configurações serial e ideal apresentadas na Tabela 1 e 2. Porém observa-se na configuração 6D, que o módulo ideal apresenta o menor tempo de execução e uma alta demanda no uso dos recursos do FPGA. A solução para este caso então, seria utilizar a configuração parcialmente ideal 6A. Uma outra possível solução seria a aquisição de um FPGA com mais recursos lógicos disponíveis para o processamento em questão.

Vê-se claramente que algumas aplicações apresentarão muitas outras configurações que podem ser exploradas, no intuito de apresentar um melhor desempenho em termos de tempo de execução, uso de recursos e custos desejáveis.

O ganho produzido pelo uso do paralelismo associado à computação científica apresentado pela arquitetura proposta, também pode ser observados analisando os resultados apresentados na Tabela 4. Este ganho vem associado ao desempenho em termos de tempo de execução e consequentemente ao consumo dos recursos disponíveis no FPGA. Comparando a Tabela 4 com os recursos lógicos utilizados para cada configuração, pode-se concluir que o uso das configurações módulo ideais desenvolvidos na arquitetura proposta oferece vantagem em relação às configurações seriais e parcialmente ideais no aspecto tempo de execução, uma vez que a configuração módulo ideal apresenta melhor desempenho neste sentido, apesar de que para a maioria das configurações, o mesmo consumiu uma quantidade de recursos lógicos maior do que nas demais configurações. Ao implementarmos os módulos usando paralelismo e com a possibilidade do uso da configurabilidade, houve redução significativa no tempo de execução das operações executadas pelos módulos desenvolvidos.

TABELA IV : *SPEED*UP DAS CONFIGURAÇÕES SERIAL/IDEAL E SERIAL/PARCIALMENTE IDEAL

Caso Aritmético	Speed up: serial/ ideal
1B	2,00
1C	2,50
2B	2,00
2C	2,00
3B	1,80
3C	3,46
3D	9,00
4B	1,93
4C	4,16
4D	4,22
5B	5,27
5C	8,06
5D	12640
6B	2,08
6C	7,71
6D	987,43

De acordo com a Tabela 4 os maiores ganhos foram obtidos nas operações de multiplicação matricial e principalmente nas multiplicações matriciais intervalares, conforme apresentados nos casos aritméticos 5 e 6. Observa-se então a necessidade da análise dos requisitos de cada aplicação, antes que seja feita a escolha da configuração do *hardware.*

VI. CONCLUSÕES

Este trabalho propôs uma arquitetura paralela reconfigurável em FPGA para implementação de operadores elementares da aritmética intervalar. Essa arquitetura foi projetada com o objetivo de diminuir o tempo de processamento das aplicações que empregam rotinas baseadas em aritmética intervalar, uma vez que ela pode proporcionar tempo de resposta menor do que os programas desenvolvidos em *software* empregando um paradigma de programação sequencial.

A utilização de técnicas de paralelismo possibilitou a redução no tempo de execução das operações intervalares realizadas pelos módulos e a utilização de técnicas de reconfigurabilidade tornou a arquitetura proposta mais flexível, no que diz respeito ao tempo de execução e à demanda de recursos lógicos do FPGA. Os resultados de simulação pós-síntese comprovam o aumento de desempenho e a flexibilidade da arquitetura proposta. Assim, considerando a arquitetura de *hardware* reconfigurável paralela proposta e projetada, para implementação de operadores elementares da aritmética intervalar e os resultados obtidos, conforme apresentado na seção V, pode-se concluir que os objetivos propostos foram alcançados completamente. A arquitetura atende os requisitos como desempenho, flexibilidade, tempo de resposta e custos desejáveis da carga (requisitos não funcionais), que podem variar de acordo com a aplicação de carga de trabalho.

O uso da reconfiguração permite a tomada de decisão quando da escolha do FPGA, uma vez que a arquitetura apresentada possui escalabilidade em função dos recursos disponíveis no dispositivo. Deve-se ressaltar a importância da configuração escolhida, de modo que a mesma seja a mais adequada, uma vez que algumas aplicações requerem altas taxas de processamento, outras aplicações, priorizam custos de implementação. Sendo assim, uma configuração adequada permite a implementação de aplicações com alta demanda de recursos, de acordo com os recursos de hardware disponíveis. Com a utilização da computação reconfigurável e do paralelismo, pode-se fazer as operações definidas pela reconfiguração de forma simultânea, podendo assim obter um ganho no tempo de processamento. Caso haja necessidade de economia em recursos lógicos, pode-se optar por diminuir o grau de paralelismo.

As principais contribuições desta pesquisa foram: a proposta e o projeto da arquitetura de *hardware* paralela reconfigurável modular em FPGA, para implementação de operadores elementares da aritmética intervalar possibilitando o aumento da flexibilidade e reduzindo o tempo de execução dos operadores da aritmética intervalar; a análise do desempenho obtido com as diferentes configurações implementadas e simuladas com configurações variadas, utilizando linguagem de descrição de *hardware* no ambiente de simulação da *Xilinx*

Os trabalhos relacionados que mais se aproximam desta pesquisa foram os descritos em [13] e em [29]. Nossa arquitetura difere destas porque é composta de uma biblioteca de operações elementares da aritmética intervalar, através de módulos de operações elementares otimizados. Sendo assim, ao selecionar, parametrizar, interligar estes módulos básicos e/ou os módulos ideais, a arquitetura reconfigurável proposta implementará os operadores intervalares, acelerando os algoritmos da aritmética intervalar, diminuindo o tempo de processamento. Esta arquitetura pode ser usada em qualquer aplicação que envolva operações intervalares.

Não foram encontrados na literatura pesquisada, trabalhos que utilizassem das vantagens da computação reconfigurável em conjunto com as técnicas de paralelismo, as quais foram as principais responsáveis pela diminuição do tempo de execução das operações realizadas pela arquitetura desenvolvida, na aceleração de operações aritméticas intervalares. Isto impossibilitou estabelecer comparações entre a arquitetura proposta e as outras arquiteturas existentes na literatura. Contudo, pode-se verificar que em relação à arquiteturas tradicionais da aritmética intervalar, com a arquitetura implementada, obteve-se ganhos significativos.

É interessante ressaltar também que em consulta à lista de pesquisadores da aritmética intervalar (*Reliable Computing*), houve retorno de trabalhos que estudam e buscam otimizar aplicações que usam aritmética intervalar, mas nenhum que busca otimizar os algoritmos da aritmética intervalar. Talvez, justamente por esses algoritmos apresentarem um elevado tempo de processamento, sendo inviável a sua implementação, sem a utilização de técnicas de paralelismo. Os organizadores desta lista de discussão *Reliable Computing* demonstraram então, interesse nos resultados desta pesquisa.

Esta pesquisa cria possibilidades para a realização de diversos trabalhos futuros, uma vez que na literatura existem alguns trabalhos que utilizam aritmética convencional em *hardware*. Não foi encontrado nenhum trabalho que utiliza aritmética intervalar em *hardware*, empregando conceitos de paralelismo computação reconfigurável e principalmente uma arquitetura modular em *hardware* utilizando técnicas de reconfiguração dinâmica e parcial.

REFERÊNCIAS

[1] PITAC Computational Science: Ensuring America's Competitivene. National Coordination Office for Information Tecnology Research & Development. 4201 Wilson Boulevard, Suite II - 405. Arlington. 2005.

[2] R. H. Landau, R. Wangberg, K. Augustson, M.J. Páez, C. C. Bordeianu, C. Barnes. A First Course in Scientific Computing. Copyright c 2005 by Princeton University Press. Princeton and Oxford. 41 William Street, Princeton, New Jersey 08540 3 Market Place, Woodstock, Oxfordshire OX20 1SY.

[3] R. M. P Trindade. Uma Fundação Matemática para Processamento Digital de Sinais Intervalares. Natal, RN. Universidade Federal do rio Grande do Norte: Centro de Tecnologia Programa de Pós Graduação em Engenharia Elétrica e de Computação. Tese de Doutorado, 2009

[4] R. Moore. Dimensional Analysis. E. Cliffs, Ed. New Jersey: Prentice Hall, 1966.

[5] L. A. Zadeh. Fuzzy sets. Information and Control, Prentice Hall PTR Upper Saddle River, NJ, USA, v. 8, n. 3, p. 338{353, 1965. ISSN 00199958.

[6] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems, v. 100, p. 9{34, 1999. ISSN 01650114.

[7] Systems, v. 100, p. 9{34, 1999. ISSN 01650114.Silveira, M. M. M. T. Teoria Fuzzy Intervalar: Uma Proposta de Integração da Teoria Fuzzy à Computação Intervalar. Master's thesis, UFRN, Natal - RN, 2002.

[8] R.E. Moore. Interval Arithmetic and Automatic Error Analysis in Digital Computing. Tese de Doutado, Department of Mathematics, Stanford University, Stanford California. 1962. NR – 0440211.

[9] C. P. Callender, C. F. N. Cowan. Implementation of Numerically Robust Fast Recursive Leat Squares Adaptive Filters Using Interval Arithmetic. ELECTRONICS LETTERS 26th March 1992 Vol. 28 No. 7.

[10] Garrozi. A Aritmética Intervalar como Ferramenta para a Solução de Problemas de Computação Científica. In: Revista Eletrônica de Iniciação Científica - Sociedade Brasileira de Computação, Artigos de pesquisa e Desenvolvimento, ano II, 2002, n 1, Março.

[11] A. W. Lodwick. A Comparison of Interval Analysis Using Constraint Interval Arithmetic and Fuzzy Interval Analysis Using Gradual Numbers. Fuzzy Information Processing Society, 2008. NAFIPS 2008. IEEE. Annual Meeting of the North American Digital Object Identifier: 10.1109/NAFIPS.2008.4531302 Publication Year: 2008, Page(s): 1 – 6.

[12] C. A. Holbig. Ambiente de Alto Desempenho com Alta Exatidão para a Resolução de Problemas. 2005. Tese (Instituto de Informática/Programa de PósGraduação em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre.

[13] A. Amaricai, M. Vladutiu, L. Prodan, M. Udrescu. Design of Addition and Multiplication Units for High Performance Interval Arithmetic Processor. Advanced Computing Systems and Architectures (ACSA) Research Group, Computer Science and Engineering Department, "Politehnica" University of Timisoara.IEEE, 2007. DDECS '07. IEEE DOI: 10.1109/DDECS.2007.4295285.

[14] R. B. Kearfott. Interval Computations: Introduction, Uses, and Resources. Department of Mathematics University of Southwestern Louisiana U.S.L. 1996. Box 4-1010, Lafayette, LA 70504-1010 USA email: rbk@usl.edu.

[15] R. E. Moore, R. B Kearfott, M. J Cloud. Introduction to Interval Analysis. SIAM (Society for Industrial and Applied Mathematics Philadelphia). 2009. ISBN 978-0-898716-69-6.Page(s): 19-28 and 157-170.

[16] G. Shan, Y. Baoming, Y. Qiang. Interval Analysis and Its Applications to Control Problems. Education and Information Technology, 2010 International Conference. Vol. 1. IEEE Conference Publication. Page(s): V1-213 - V1-218. DOI 10.1109/ICEIT.2010.5607749.

[17] J. Ninin, F. Messine, P. Hansen. A Reliable Affine Relaxation Method for Global Optimization. October 23, 2012. ed: Montréal : Groupe d'études et de recherche en analyse des décisions, 2010. Cahiers du GÉRAD, G-2010-31.

[18] A. F. Finger, A. B. Loreto, M. A. Campos, F. R. G. Varjão, M. G. dos Santos. The Computational Complexity of Problems to Compute Intervals Wrappers for Random Variables Uniform, Exponential and Pareto. 978–1–4673–0793–2/ c 2012 IEEE

[19] F. T. Santana, F. L. Santana, A. D. D. Neto, R. H. N. Santiago. Sinais e Sistemas Definidos Sobre Aritmética Intervalar Complexa. Tema: Tendências em Matemática Aplicada e Computacional, 13, Nº. 1 (2012), 85-96. DOI: 10.5540/tema. 2012.013.01.0085. © Uma Publicação da Sociedade Brasileira de Matemática Aplicada e Computacional.

[20] E. J. Rothwell, J. Edward, M. J. Cloud. Automatic Error Analysis Using Intervals. – IEEE Transations on Education, vol.55, nº 1, February 2012.0018-9359. IEEE DOI 10.1109/TE.2011.2109722.

[21] Z. Zhou, Z. Liu, B. Zeng, Y. Pang, L. He. Application of the Interval Arithmetic in Reliability Analysis of Distribution System. Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2012 International Conference on. IEEE, 2012.

[22] K. Compton, S. Hauck. Reconfigurable Computing: A Survey of Systems and Software, *ACM Computing Survey*, 2002. Vol. 34, n. 2, p. 171-210

[23] D. Patterson, J. Hennessy. Computer Organization and design: the hardware/software interface. [S.I.]: Morgan Kaufmann Pub, 2009.

[24] B. Harikrishna, S. Ravi. Intelligent Systems and Control (ISCO), 2013 7th International Conference on Digital Object Identifier: 10.1109/ISCO.2013.6481160 Publication Year: 2013, Page(s): 265 – 270. IEEE Conference Publications.

[25] P. Bleisch, D. Morgan, D. Youngwith. D. The State of ReConfigurable Computing, 1995.

[26] I. Shiliarova, A.B. Ferrari. Introdução à computação reconfigurável. Portugal: Revista DETUA, 2003.

[27] A. Dehon. The Density Advantage of Configurable Computing. In: IEEE Computer. Vol. 33, No. 4. 2000. Page(s): 41 – 49.

[28] C. A. P. S Martins, E. D. M. Ordonez, J. B. T. Corrêa, M. B. Carvalho. Computação Reconfigurável: Conceitos, Tendências e Aplicações. In: XXII Jornada de Atualização em Informática (JAI) - Livro Texto. Campinas: Sociedade Brasileira de Computação, 2003. v. XXII, p. 339-388.

[29] R. Shettar, R. M. Banakar, P. S. V. Nataraj. Implementation of Interval Arithmetic Algorithms on FPGAs. International Conference on Computational Intelligence and Multimedia Applications 2007. 0-7695-3050-8/07 © 2007 IEEE DOI 10.1109/ICCIMA.2007.60196.

[30] H. Ratsche, R. Rokne. New Computer Methods for Global Optimization. Ellis Horwood, 2007.

[31] J. Villasenor, W. H. Mangione-Smith. Configurable Computing, *Scientific American*, Junho 1997. p. 54-59.

[32] S. Hauck, A. Dehon.Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation, Morgan Kaufmann/Elsevier, 2008.

[33] L. Bauer, C. Braun, M.E. Imhof, M.A. Kochte, Zhang Hongyan, H. Wunderlich, J. Henkel. Adaptive Hardware and Systems (AHS), 2012 NASA/ESA Conference on Digital Object dentifier: 10.1109/AHS.2012.6268667.Publication Year: 2012, Page(s): 38-45 Cited by 1. IEEE CONFERENCE PUBLICATIONS.

[34] D. Llamocca, M. Pattichis, Circuits and Systems for Video Technology, IEEE Transactions on Volume: 23, Issue: 3.Digital Object Identifier: 10.1109/TCSVT.2012.2210664 Publication Year: 2013, Page(s): 488 – 502. IEEE JOURNALS & MAGAZINES

[35] R. Harold. Review of "Highly parallel computing".Lorin IBM Thornwood Education Center, Thornwood, New York. Journal. IBM Systems Journal, archive. Vol 29. Issue 1. January 1990. Pg. 165 -166. IBM Corp. Riverton, NJ, USA.

[36] R. Kumar, D. Marinov, D. Padua, M. Parthasarathy, S. Patel, D. Roth, J Torrellas, J. Parallel Computing Research at Illinois the UPCRC Agenda. (2008).

[37] J. Castillo, J. L. Bosque, E. Castillo, P. Huerta, J. Martinez. Hardware accelerated Monte Carlo financial simulation over low cost FPGA cluster. Parallel and Distributed Processing Symposium, Internacional, vol.0. pgs. 1 – 8, 2009.

[38] Leandro, A.J.M. Explorando Linhas de Execução Paralelas com Programação Orientada por Fluxo de Dados. Tese. Universidade Federal do Rio de Janeiro. UFRJ/COPPE . Outubro 2011..

[39] R. Shettar, R. M. Banakar, P. S. V. Nataraj. Design and Implementation of Interval Arithmetic Algorithms. International Conference on Industrial and Information Systems, ICIIS 2006, 8 - 11 August 2006, Sri Lanka. 1-4244-0322-7/06/ c2006IEE.

[40] G.C. P Duarte; B.R.C. Bedregal. Introdução à Matemática Financeira. Intervalar: Análise Intervalar de Investimentos. Natal, RN. Sociedade Brasileira de Matemática Aplicada e Computacional. Depto de Informática e Matemática Aplicada, CCET, UFRN, Março, 2009.

[41] J. Hormigo; J. Villaiba; E. Zapata. Arithmetic Unit for the Computation of Interval Elementary Functions. Dept. of Computer Architecture University of Malaga. DOI: 10.1109/EURMIC.1999.794447. 1999, p: 63 - 66 vol.1. IEEE CONFERENCE PUBLICATIONS.

[42] M. Shahbazi; P. Poure; S. Saadate;M. R. Zolghadri.Fault Tolerant Five Leg Converter Topology With FPGA Based Reconfigurable Control. Industrial Electronics, IEEE Transactions on Volume: 60, Issue: 6 Digital Object Identifier: 10.1109/TIE.2012.2191754 Publication Year: 2013, Page(s): 2284 – 2294. IEEE Journals & Magazines.

[43] Benhamou; W.J. Older; A. Vellino. Constraint Logic Programming on Boolean, Integer and Real Intervals. Journal of Symbolic Computing. May 10. 1994. (Submitted).

[44] B. Catazaro; B. Nelson.Higher Radix Floating-Point Representations for FPGA-Based Arithmetic. Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines.2005. Page(s): 161 – 170.