Número de inscrição:12345
Área Temática: Área Temática

Simulação rápida e precisa de MPSoCs usando as instâncias EC2 F1 da Amazon

***Resumo*—As redes intrachip possuem uma arquitetura de comunicação escalável e de alto paralelismo para MPSoCs. Porém, os recentes avanços das tecnologias de fabricação dos circuitos integrados, tornam-se cade vez maior o número de falhas durante a confecção e operação desses circuitos. Novos métodos de verificação e correção de falhas em tempo real são implementados, porém possuem uma simulação lenta e custosa. Esse trabalho propõe uma nova metodologia de simulação acelerada por FPGA em MPSoCs, sobretudo os que utilizam métodos resilientes a erro. Além disso o novo método utilizará um serviço em nuvem, afim de acelerar e otimizar a simulação.**

***Palavras-chave*—Redes-em-Chip, tolerância a falhas, simulação, FPGA, desempenho, nuvem.**

I. Introdução

Com o crescimento da tecnologia nos sistemas computacionais modernos, cada vez mais surgem um maior número e complexidade de tarefas nas aplicações, sendo requerido desses sistemas mais funcionalidades e maior velocidade nos dispositivos de hardware que realizam tais tarefas.

Isso acontece devido a necessidade da realização de várias aplicações em um mesmo dispositivo, podendo assim atuar vários elementos de processamento, em inglês Processing Elements (PEs) em um mesmo circuito. Vale ressaltar que a evolução da tecnologia de fabricação de circuitos integrados CMOS (complementary metal-oxide-semiconductor), através da miniaturização dos transistores, tornaram esses circuitos capazes de comportar sistemas complexos.

Uma vez que o sistemas de circuitos integrados são compostos por pastilhas em que são inseridos os elementos que os compõem [2]. O fato de podermos colocar vários elementos de processamento (memórias,processadores e até placas de vídeo) torna possível a elaboração de um sistema computacional completo em um chip, em inglês Systems-on-Chip(SoCs) [1]. Por existirem diversos componentes de processamento dentro dos SoCs, torna-se necessário uma comunicação eficiente entre esses elementos, para evitar latência ou caminhos longos ou críticos de determinadas computações. Por isso é introduzido redes de interconexões em chip, também conhecidas como Network-on-Chip (NoC). A comunicação entre os elementos do SoC e a NoC é feita por meio de adaptadores de rede Basicamente uma dada NoC é constituída por roteadores e links, a partir daí é configurada como uma rede de roteadores interligados que utiliza conceitos de redes de computadores e sistemas distribuídos para as conexões intrachip dos PEs.

Utilizando redes como arquitetura de comunicação entre os PEs. As NoCs permitem uma maior escalabilidade e produção de desempenho requerido por diversos SoCs com alto grau de paralelismo [3]. Entretanto, a topologia inicial da NoC é um modelo ideal, pois essa miniaturização que reduz a área e consumo de energia do circuito integrado também o torna suscetível a falhas causadas por interferências, por desgastes devido ao envelhecimento do circuito ou falhas no processo de fabricação [4].

Por isso é necessário considerar NoCs podem possuir topologias irregulares e mesmo assim, atender os requisitos de desempenho e funcionalidades. Para isso é necessário um gerência de falhas de modo que seja realizado uma identificação dessas falhas, além de uma reconfiguração de topologia nesse tipo de NoCs.

De acordo com o [5] para um bom funcionamento de uma NoC com topologia irregular é necessário que alguns pontos sejam atendidos por essa rede intrachip. O primeiro ponto seria o algoritmo de roteamento, o mecanismo de implementação e por último o procedimento de reconfiguração.

O método de reconfiguração é de suma importância para a recuperação do sistema, considerando o tempo de recuperação, possibilidades de nós alcançáveis na rede e o desempenho da comunicação entre PEs. Um algoritmo clássico se baseia em reconfiguração estática de NoCs é proposto em [6], que demanda de um certo tempo de recuperação do SoC, o que pode ser crucial em sistemas de tempo real. Por isso, o trabalho [5], propõe uma mudança significativa no procedimento de reconfiguração visando a continuidade do funcionamento da NoCs que possuem topologias irregulares, possibilitando um maior desempenho.

Vale ressaltar que nesse trabalho foi utilizado o conceito de pré-processamento, que busca uma maior otimização da reconfiguração em tempo, uma vez que os sistemas de tempo real possuem requisitos de continuidade de funcionamento. Por mais que esse trabalho tenha trazido uma maior eficiência na previsão de falhas e correção em tempo real, a simulação a nível de computação em ciclo de clock é um gargalo em tempo para NoCs com tamanhos mais extensos e com um grande conjunto de dados ou requisições, uma vez que é um sistema robusto e suas atividades combinacionais possuem um certo tempo de computação. Diante disso esse projeto baseia-se na replicação do trabalho [5], em um circuito reconfigurável utilizando a arquitetura conhecida como NoC Phoenix [7]. Buscando uma maior eficiência na simulação nesse novo

contexto, além de uma maior assertividade na eficiência do projeto utilizando métricas de tempo, energia e latência. Que são de suma importância para esses sistemas. Para isso, foi utilizado o Framework MIDAS que tem como principal função otimizar simulações de projeto RTL utilizando FPGAs. Além disso, para um maior eficácia e escalabilidade esse sistema será portado na nuvem da Amazon através do serviço AWS e das instâncias EC2 F1.

II. Detalhamento Técnico

NoC Phoenix [7] é uma arquitetura de PEs intrachip, podendo atuar sobre uma plataforma de processadores em inglês MultiProcessor SoC (MPSoC), que utiliza uma topologia com malha 2D como infraestrutura de comunicação, sua divisão consiste em uma parte hardware chamada HwPhoenix (roteadores) e uma parte software chamada OsPhoenix (sistema operacional de cada PE) (Figura1). Basicamente a Phoenix possui uma topologia de dimensões m × n roteadores . Esses roteadores são conectados entre si por meio de links bidirecionais e cada roteador possui cinco portas: Norte, Sul, Leste, Oeste e Local(interface de rede).

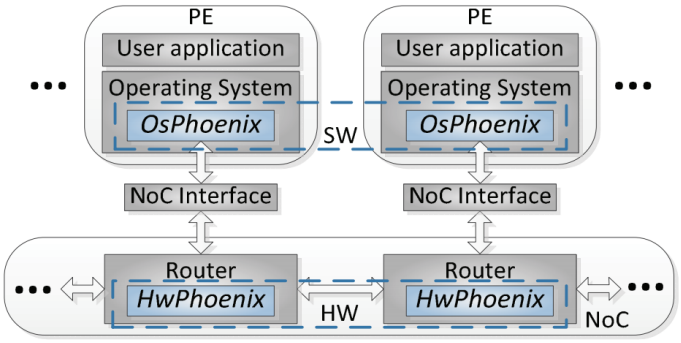

Fig. 1: Arquitetura básica Phoenix (contendo HwPhoenix e OSPhoenix)

Para realizar uma significativa melhora na eficiência de uma NoC, sobretudo na reconfiguração com topologia irregulares, no trabalho [5] realizado uma Reconfiguração no OSPhoenix e pré-processamento de cenários. O OSPhoenix é responsável por gerenciar os algoritmos de roteamento e assim carregar as tabelas em um dado cenário, por isso é de suma importância que haja uma eficiência na identificação e tratamento em cenários de falha e assim a utilização do algoritmo de roteamento mais eficaz. O pré-processamento realizado encima da Phoenix busca ser assertivo em relação a possíveis falhas para um sistema com variados cenários , e quanto mais cenários processados maior é a cobertura de falhas. O pré-processamento de uma grande quantidade de cenários é um problema muito complexo, pois consome muito tempo e memória. Por isso foram utilizadas técnicas de cobertura de cenários,nível de similaridade de cenários e redução de cenários de cobertura.
Para a validação do desse pré-processamento, é necessário uma comparação sobretudo em tempo e assertividade de vários cenários. Por isso é requerida uma simulação robusta com diferentes tipos requisições e tamanhos de rede, para assim validar a eficiência dessa importante mudança na NoC. Contudo, vale ressaltar que simulações com grandes cargas de dados realizadas a nível de ciclo em softwares de microarquitetura são demasiadamente lentas, o que dificulta a validação de sistemas reais. Uma alternativa a isso seriam as metodologias de amostragem de simulação, que trabalham com grandes quantidades de dados de forma mais otimizada, porém essas técnicas não apresentam limites de erros, o que pode gerar avaliações erradas do sistema em validação [9]. Diante disso esse trabalho pretende utilizar uma nova ferramenta de simulação e consequentemente validação da NoC Phoenix com pré-processamento, de forma mais eficaz. Para tal será utilizado uma metodologia conhecida como MIDAS [8].
O MIDAS é um framework de simulação que gera automaticamente um simulador acelerado po FPGA (field programmable gate arrays) a partir de projetos que utilizam o conceito de register-transfer level (RTL), que são projetos que utilizam linguagens de descrição de hardware (HDL) para representar circuitos de mais baixo nível, assim simplificando a implementação da lógica dos projetos. Basicamente o MIDAS possui dois principais agentes o host e alvo, o alvo é a máquina sendo simulada (por exemplo um SoC), já o Host a máquina que executa(alvos) a simulação. Esse host é um FPGA híbrido (FPGA + CPU embutido). A arquitetura interna do framework MIDAS pode ser vista na figura 2, para realizar a transformação de um projeto RTL em um simulado acelerado por FPGA é necessária a ação de um compilador FIRRTL (Flexible Intermediate Representation for RTL), que é uma representação intermediária para circuitos digitais projetados como uma plataforma para escrever transformações em nível de circuito, FIRRTL representa o circuito elaborado e padronizado que o Chisel HDL produz. Além disso, o FIRRTL representa o circuito imediatamente após a elaboração de Chisel mas antes de qualquer simplificação do circuito [10]. Dentro do compilador FIRRTL, contemos alguns passos:

- **Macro Mapping (opcional)** : Mapeamento de macro mapeia blocos de memória independentes de tecnologia a blocos macro dependentes de tecnologia para estimativa de energia.
- **FAME1 Transform**: A transformação FAME1 separa o clock de destino do clock do host anexando sinais de ativação a todos os elementos de estado. Isso permite que partes do simulador parem quando não estão prontas, permitindo que o simulador tolere latências variáveis na plataforma host e assegurando que as simulações sejam determinísticas.
- **Scan chains (opcional)**: São capturas de estados instantâneos(snapshots) para voltar a realizar na simulação em software a nível RTL para estimação de potência.
- **Simulation Mapping**: O mapeamento de simulação envolve o design de destino transformado inserindo canais de token de tempo / buffers de rastreio. O resultado é um

módulo de simulação independente de plataforma para simulação baseada em tokens.

- **Platfom Mapping**: O último passo do compilador MIDAS, liga todos os módulos de simulação hospedados no FPGA com lógica específica da plataforma.

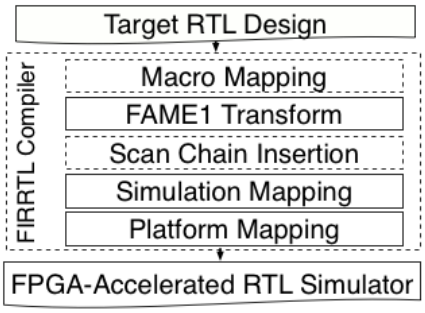

Fig. 2: Passos personalizados do compilador para gerar um simulador RTL acelerado por FPGA

Diante disso, a principal função do MIDAS é gerar um simulador de alto desempenho em um alvo a partir de projeto RTL, esse mapeamento pode ser visto na Figura 3.

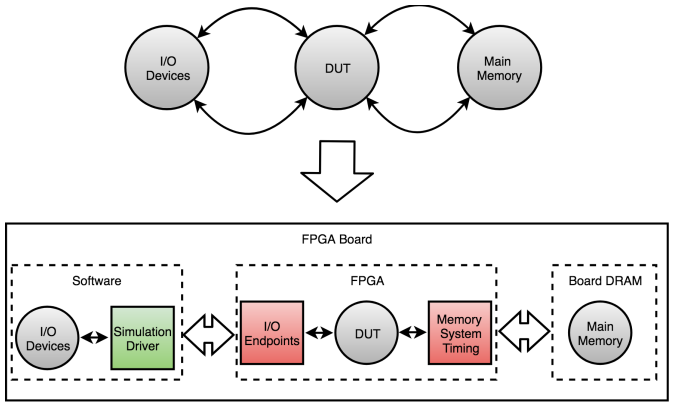

Fig. 3: Mapeamento do projeto alvo para a plataforma de FPGA host - Os dispositivo **I/O - Devices** são modelados em software e operam ao lado do **driver de simulação**, pois as injeções de dados são mais raros. Em geral, a comunicação realizada entre a um modelo de software do host e um modelo RTL alvo possui uma sobrecarga em tempo, por isso existe uma otimização através de **I/O- endpoints** , utilizando tokens de tempo que em alto nível realizam a comunicação com com o **DUT** (device under test) utilizando todo o desempenho de velocidade dos periféricos do FPGA. Dentro do alvo existe um **modelo de temporização** que tem a função de memória de acesso à **memória principal**, essa por sua vez está no host.

Uma vez que temos todo o arcabouço para realizar uma simulação rápida em MPSoCs é necessário utilizar ferramentas adequadas para tal, de forma que os requisitos principais dessa simulação sejam atendidos. Com isso é necessário o uso de FPGA com alta velocidade e capacidade de processamento. Esse tipo de dispositivo não é tão acessível, devido ao seu alto custo financeiro. Diante disso, esse trabalho utilizará os recursos de uma FPGA com essas especificações entre outras vantagens, como um serviço no ambiente de nuvem Amazon Web Service(AWS).

A AWS é um plataforma de serviço em nuvem que dispõe de diversos serviços de infra-estrutura, possibilitando usuários ou empresas a utilização desses serviços. A plataforma oferece poder computacional, armazenamento em nuvem distribuição de conteúdo e outros serviços, cada um com sua funcionalidade nas áreas em que são necessários. Dentre esses serviços, existe o EC2F1 que é utilizado para o processamento de dados remoto em FPGAs. O EC2(Amazon Elastic Compute Cloud) é um serviço que fornece acesso aos servidores que possuem os recursos computacionais (essa gerência é realizada sob demanda) , assim o uso do serviço se torna escalável. Um dos tipos de instância EC2 é a EC2 F1, o Amazon EC2 F1 é uma instância de computação com FPGAs que podem ser programadas para criar acelerações de hardware personalizadas para aplicações. Existem outras funcionalidades que essa instância possui, porém este trabalho busca alto desempenho computacional que EC2 F1 disponibiliza. Esse poder de processamento é possível, sobretudo pelas FPGAs que são disponibilizadas na EC2 F1. Essas são Xilinx UltraScale+ VU9P fabricadas em tecnologia de 16 nm, contendo 1.182 LUTs, 2.364 flip-flops, 345.9 Mb de memória e 832 de I/O. Vale ressaltar que essa FPGA é caracterizada por possui as mais altas performances entre os dispositivos produtos da Xilinx. Toda a estrutura disponível pela instância, possui uma estratégia de comunicação para realizar a aceleração baseada em FPGA, isso se dá através de conexões via PCIe (PCI Express) com os elementos de CPU (pós-processadores).

Para todo o processo desenvolvimento de um core AWS, é necessário realizar essas tarefas (figura 4):

1) Desenvolver a Lógica Customizada (Custom Logic - CL), que é basicamente o projeto RTL combinado com as ferramentas de desenvolvimento disponíveis na AMI (Amazon Machine Image) e o HDK (Hardware Development Kit).
   a) FPGA Deveploper AMI: Fornece as informações necessárias para executar uma instância, que é um servidor virtual na nuvem.
   b) Kit de desenvolvimento - HDK: O AWS FPGA HDK é o kit oficial fornecido pela AWS para facilitar o desenvolvimento de RTL (Verilog / VHDL) de uma imagem FPGA da Amazon (AFI).
2) Combinar a CL com uma lógica fornecida pela AWS chamada de FPGA Shell.
   a) FPGA SHELL: É a lógica de plataforma AWS responsável por cuidar dos periféricos externos FPGA, PCIe, DRAM e Interrupções.
3) Criar uma Amazon FPGA Image (AFI) a partir da CL construída.
   a) AFI: AFI (Amazon FPGA Image) é uma imagem pronta pra ser instanciada na F1, o desenvolvedor pode registrar uma AFI para ser utilizada várias

vezes por um uma F1 e/ou pode ser utilizada por várias instâncias F1.

4) Carregar a AFI gerada em uma FPGA conectada à instância F1 ou disponilizar no AWS Marketplace.
   a) AWS Marketplace: O AWS Marketplace é uma loja online que ajuda os clientes a descobrir, comprar, migrar e começar a usar imediatamente o software e os serviços de que precisam para criar produtos e gerir suas empresas.

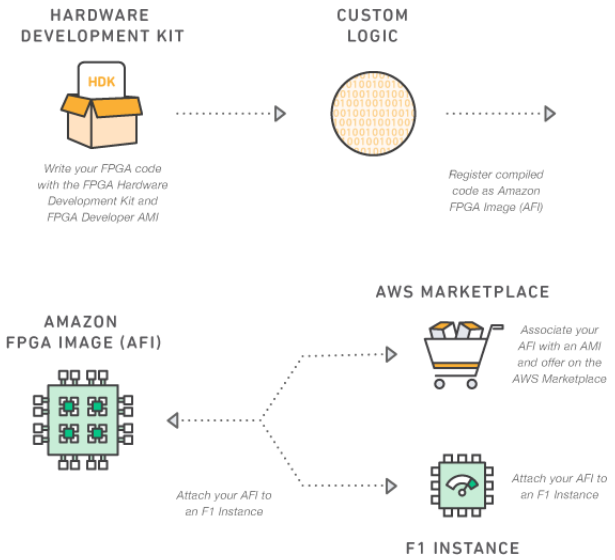

Fig. 4: Processo de desenvolvimento de cores utilizando o serviço EC2 F1 a partir de um projeto RTL

III. Objetivos

Tendo em vista o cenário de oportunidades de pesquisa com as ferramentas explicitadas na seção anterior, enumeramos abaixo os objetivos gerais e específicos desta pesquisa.

- Objetivos Gerais

  1) Propor uma nova metodologia de simulação rápida de MPSoCs, sobretudo do ponto de vista de suas interconexões.
  2) Implementar o modelo proposto em um NoC que utiliza pré-processamento, caracterizando um modelo tolerante a falhas.
  3) Realizar a implementação desse serviço e aceleração do mesmo, utilizando o serviço AWS da Amazon.

- Objetivos Específicos
  1) Estudar o modelo da NoC Phoenix com reconfiguração e pré-processamento, além de implementar em uma linguagem a nível RTL suportada pelo MIDAS.
  2) Fazer estudo sobre arquitetura MIDAS e outros métodos de aceleração de simulação.
  3) Realizar um estudo do serviço AWS e toda a arquitetura em nuvem necessário para implementação de cores em FPGAs remotos.

De acordo com os objetivos pré-estabelecidos, segue abaixo uma tabela que descreve as atividades propostas a serem realizadas nesse projeto, bem como a estimativa de tempo de cada uma.

Tabela I: TABELA DE ATIVIDADES

Atividade	**Estimativa de Tempo**
Revisão de literatura sobre NoCs e NoC Phoenix	Mês 1 e 2
Revisão de literatura sobre da Phoenix resiliente a erros, utilizando pré-processamento,além de sua implementação em VHDL/Verilog	Mês 3
Revisão de literatura sobre MIDAS e outras aceleradores de simulação	Mês 4
Revisão de literatura sobre o serviço AWS e utilização da EC2 F1	Mês 5
Implementação da Medologia de simulação rápida em MPSoCs	Mês 6 a 12
Testes do Simulador na NoC phoenix, resiliente a erros, utilizando pré-processamento, bem como a implementação no ambiente de nuvem	Mês 13
Qualificação	Mês 14
Comparação dos resultados obtidos com sistemas já existentes	Mês 15 e 16
Escrita para congresso	Mês 17
Escrita da dissertação	Mês 18 a 20

IV. Resultados Esperados

Tendo em vista que as simulações de NoCs com uma maior complexidade (inclusão de técnicas) leva muito tempo para realizar simulações e consequentemente diagnósticos. Esse muitas vezes não é medido de forma mais rigorosa em projetos RTL com simuladores comuns, porém já se sabe que é inviável para uma maior escalabilidades de PEs. Por isso esse pretendemos fazer uma análise do desempenho atual de simulação desse projeto citado, além de propor uma nova solução de simulação eficiente utilizando as ferramentas que foram citadas nesse documento. Por fim, para tornar viável metodologia , se faz necessário a comparação com vários simuladores e com diversos tipos de MPSoCs, tornando a proposta viável em outros cenários de uso.

REFERÊNCIAS

[1] PASRICHA, Sudeep; DUTT, Nikil. On-Chip Communication Architectures: System on chip interconnect. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[2] T. E. Kolding, "A four-step method for de-embedding gigahertz on-wafer CMOS measurements," in IEEE Transactions on Electron Devices, vol. 47, no. 4, pp. 734-740, Apr 2000.

[3] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger and Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 1, pp. 3-21, Jan. 2009.

[4] Jerraya, A. A.; Wolf, W. "Multiprocessor Systems-on-Chips". Em: Morgan Kaufmann Publishers Inc, 2005.

[5] SILVEIRA, J. et al. Scenario preprocessing approach for the reconfiguration of fault-tolerant NoC-based MPSoCs. Microprocessors and Microsystems, v. 40, p. 137–153, 2015.

[6] STRANO, A. et al. OSR-Lite: Fast and Deadlock-free NoC Reconfiguration Framework. International Conference on Embedded Computer Systems (SAMOS), pp. 86-95, 2012.

[7] C. Marcon, A. Amory, T. Webber, T. Volpato, L. Poehls, Phoenix NoC: a distributed fault tolerant architecture, in: International Conference on Computer Design (ICCD), 2013.

[8] Kim, Donggyu, Christopher Celio, David Biancolin, Jonathan Bachrach and Krste Asanovic. "Evaluation of RISC-V RTL with FPGA-Accelerated Simulation." (2017).

[9] R. E. Wunderlich, T. F. Wenisch, B. Falsafi and J. C. Hoe, "SMARTS: accelerating microarchitecture simulation via rigorous statistical sampling," 30th Annual International Symposium on Computer Architecture, 2003. Proceedings., 2003, pp. 84-95.

[10] LI, Patrick S.; IZRAELEVITZ, Adam M.; BACHRACH, Jonathan. Specification for the FIRRTL Language. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-9, 2016.