2013 年 12 月 16 日 情報システム基盤学合同輪講発表資料

NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures

著者： Andreas Prodromou, Andreas Panteli, Chrysostomos Nicopoulos, Yiannakis Sazeides
出典： *Proceedings of the International Symposium on Microarchitecture (MICRO '12), pp.60-71, 2012.*
発表者： 高性能コンピューティング講座 本多・近藤研究室 1353030 松村 正隆

1 研究背景

現在のプロセッサにおける性能向上方法は、シングルコアの演算能力向上が電力等の理由から限界に達したため、マルチコア化による並列処理に変わっている。しかし、コア数が増えるにつれて各コア間を結ぶリンクは複雑化し、通信性能の低下や設計の複雑化、実装面積のオーバヘッドなどの問題が生じた。そこで近年では *Network-on-Chip(NoC)* によるコア間接続が注目されている。NoC の概略図を図 1 に示す。

個々の四角がコアを表しており、それらを線で描かれたリンクで結んでいる。NoC は各コアにルータを内蔵し、他のコアからのデータはルータを介してパケットの形で伝送される。目的のデータが隣接コア以外の場合、データは複数のルータを経由（*hop*）して伝送される。この点が従来の通信方式との最大の違いである。NoC はコア内に演算部と通信部が内蔵されたモジュール構造のため、新たにコアを追加する場合も簡単に設計できる上、コア間の接続も複雑化しないために高いスケーラビリティがある。しかし、通信に複数のルータを経由する仕組みのため、どこか 1 つのルータが故障してしまうだけでシステム全体が機能不全に陥る可能性がある。そこで、NoC の信頼性向上は今後のプロセッサの発展に不可欠と言える。

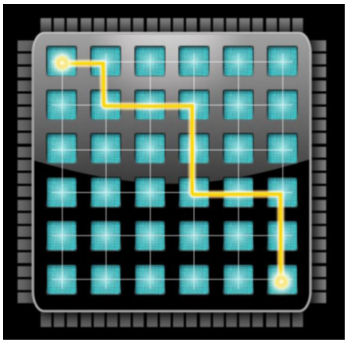

図 1: NoC 概略図

2 目的

本論文では NoC の信頼性向上のため、ルータで発生したエラーを検出するシステムを考案する。システムに求められる要件を次に示す。1) システムに影響するエラーは可能な限り検出する。2) エラー発生後はなるべく早く検出する。3) 演算能力・面積・消費電力の面で、オーバヘッドをなるべく小さくする。これらの要件を満たすシステムとして、NoCAlert を提案する。

3 NoCAlert

3.1 概要

NoCAlert はルータ内で発生したエラーを検出するシステムである。NoCAlert の大きな特徴は、NoC を構成する個々のモジュールの機能に対応した専用のエラー検出器を用いる点にある。その際、エラー検出の指標となるのがハードウェアの合法性と固有ルール侵害である。

3.2 ハードウェアの合法性と固有ルール侵害

NoCAlert で用いる概念の **"ハードウェアの合法性"** と **"固有ルール侵害"** について説明する。

ハードウェアの合法性：ハードウェアの機能から見た時に、その出力があり得るものなのかを示す。あり得ない出力がされた場合を「違法」とする。

固有ルール侵害：ハードウェアの合法性で違法とされた状態を指す。例えば NoC 内のルータ間の通信で、対象と異なるルータに対して出力を行った場合、それは「違法」であり、ルータにおける「固有ルール侵害」となる。

3.3 NoCAlert におけるエラー検出方法

NoC ルータで発生するシステムに影響するエラーは、必ず前述した固有ルール侵害を引き起こす。そのため、NoCAlert は NoC ルータを構成する全ての要素において固有ルールを調査し、それに応じたエラー検出回路を実装する。全ての構成要素はモジュール化され、階層構造になっているため、下位階層からボトムアップで固有ルールを調査していけば全ての固有ルールが求まる。実際に 8×8 のメッシュ構成の NoC を調査したところ、32 個の固有ルールが見つかった。図 2 のアービタを用いて、固有ルールとその検出回路を例示する。

アービタはルータ間で同じ経路を使うパケットが存在する場合、それらが競合しないようにパケット伝送リクエストに対して常に一つのリクエストだけパケットを流せるように調停する役割を持つ。図 2 ではリクエストがある場合を 1、ない場合を 0 とし、リクエストしたパケットが伝送される場合は 1、されない場合を 0 としている。アービタの固有ルールは 1) リクエストがないのに認証してはならない、 2) 認証できるのは常に 1 つのみ、 3) 1 つのリクエストは 1 つの認証しかできない、 4) 1 つ以上の

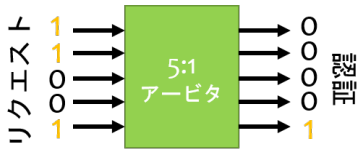

図 2: アービタ概略図

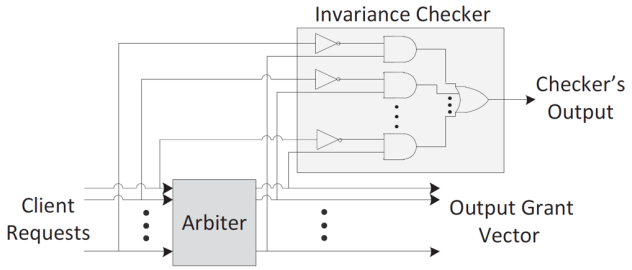

図 3: アービタの固有ルールチェック回路

リクエストがある場合 1 つの認証をしなくてはならない、の複数ある。これらの固有ルールをチェックする回路を図 3 に示す。図 3 の回路は入力の NOT と出力の AND を取り、アービタの固有ルール侵害が発生すると 1 を出力する。このように、各コンポーネントが持つ固有ルールをチェックする回路を実装してエラー検出を行う。

4 実験方法

提案手法の効果を調べるため、GARNET cycle-accurate NoC シミュレータを用いた性能測定と、Synopsys Design Compiler を用いたハードウェアの実装オーバヘッド調査を行った。比較対象として、最新のエラー検出システムである ForEVeR[1] と比較した。NoC のモデルは 8×8 のメッシュ構成で、65nm プロセスのライブラリを用いた。

想定したエラーは制御信号におけるシングルビット反転で、これを NoC ルータの全てのモジュールにおける入力、出力の両方に発生させた。これより、総エラー発生箇所は 11808 個となった。更に、ネットワークの状態によってエラーが引き起こす影響は変化すると考えられる。例えばアービタでシングルビット反転が起こっても、NoC での通信が活発な場合、他にも送信リクエストがあるのでエラーが隠ぺいできる可能性があるため、ネットワーク稼働率とエラー注入タイミングの条件も加えた。まず、ネットワークの稼働率 (node/flit/cycle) を 10%から 40%まで、5%刻みの 7 通りで変化させ、更に NoC でアプリケーションが開始された後のエラー発生のタイミングを 0 サイクル後、32K サイクル後、64K サイクル後の 3 通りとした。これより、総実験ケースは $11808 \times 7 \times 3 \approx 248K$ となった。

5 結果

図 4 に注入されたエラーがどのように検出されたのかを示す。図中の 3 種類のエラーについて説明する。

True Positive : システムに影響するエラーで、検出器が正しく検出できたもの。

True Negative : システムに影響しないエラーで、検出器が検出しなかったもの。

False Negative : システムに影響しないエラーにも関わらず、検出器が検出してしまったもの。

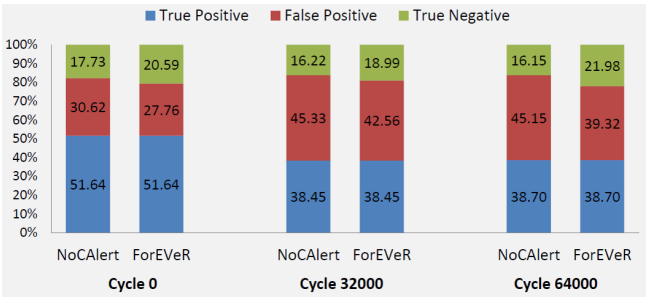

図 4: 注入されたエラーにおける検出のされ方の割合

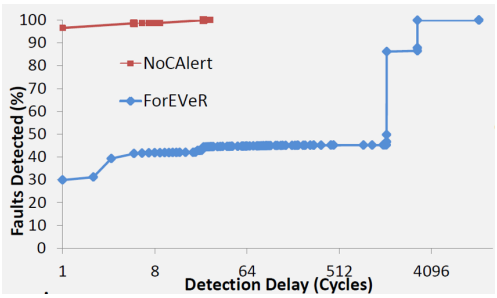

図 5: 各手法におけるエラー検出のレイテンシ

また、エラーのケースとしては「システムに影響するのに、チェッカが検出できなかったエラー。」という、いわゆる「検出漏れ」があるが、提案手法と従来手法の両方ともこのケースは 0%だった。どちらの手法もシステムに影響を及ぼすエラーは漏れなく検出できているが、提案手法は従来手法よりも True Negative の割合が僅かに少ない。これは、従来手法のエラー検知方法がある程度の時間的な区切りをつけて、その区切りをまとめてエラーチェックする方法なのが原因である。システムに影響しないエラーは時間経過とともに消滅するため、従来手法のエラーチェック方法のほうが有利となるためである。

次に、図 5 に各手法におけるエラー検出のレイテンシを示す。横軸はエラー発生から検出にかかったサイクル数で、縦軸は総エラーに対する割合である。これより、提案手法は 97%のエラーを発生と同サイクルに検出している上、全てのエラー検出を 64 サイクル以下で行っている。一方で従来手法はエラー検出に 2000 サイクル以上かけているものが 60%近くある。

最後に、オーバヘッドを調査した結果、面積オーバヘッドは 3%程度、消費電力オーバヘッドは 0.7%程度、クリティカルパスオーバヘッドは 1%程度と、極めて低いオーバヘッドで提案手法を実現できることが判明した。

6 まとめ

提案手法は従来手法と同等の検出率でありながら、高速にエラーを検出でき、オーバヘッドも小さいことがわかった。

参考文献

[1] R. Parikh and V. Bertacco., "Formally enhanced runtime verification to ensure noc functional correctness.", In Proc. of the International Symposium on Microarchitecture (MICRO), 2011.