大域ディペンダブル情報基盤プロジェクト
～ディペンダブルアーキテクチャグループ　平木研究室～

平木　敬

情報理工学系研究科コンピュータ科学専攻

あらまし　平成１５年度における平木研究室では、アーキテクチャ的アプローチによるディペンダビリティの実現を目指し（1）CPU そのものを超ディペンダブルとすることを目的とした、FPGA を用いた代用と再構成プロセッサアーキテクチャの研究と、(2)超高速ネットワークを対象とした IDS をアーキテクチャを用いて実現する方式の研究を実施した。FPGA を用いた代用と再構成プロセッサアーキテクチャの研究では、超高信頼性を実現する部分を、演算器からプロセッサ全体に拡大し、System　C 言語を用いてプロセッサのシミュレーションを実施した。超高速ネットワークを対象とした IDS アーキテクチャでは、２ポート・ネットワークインタフェースカードの試作を開始し、そこに実現するためのデータパターンマッチの基本方式を検討した。

１．はじめに

情報システムのディペンダビリティを向上させるアーキテクチャ的アプローチとしては、情報システムの中枢である CPU に超ディペンダビリティを実現することと並び、情報システム全体のディペンダビリティをアーキテクチャ技術を用いて向上させることが求められる。平木研究室では、前者の問題を解決することを目的として、平成１４年度に引き続き代用と再構成を用いた FPGA による超高信頼 CPU の研究を実施した。このテーマにおける目標は、100 万から 200 万ゲート規模の FPGA を用いることにより、構成要素の９０%の論理セルに故障が発生した場合でも、数倍の性能低下に影響をとどめ、動作を実現する CPU を実現することである。平成１５年度はその第二年度として、CPU 全体を高信頼とするための基本方式を策定するとともに、CPU システム全体を System　C 言語により記述し、シミュレーションを実施した。

後者の研究内容として、平成１５年度から新たに FPGA ベースネットワークインタフェースカードを用いた、超高速 IDS の実現方式の研究を開始した。情報システムのディペンダビリティを確保するためには、システム外部からインターネットなどを介して来る不正データ、不正な攻撃から情報システムを保護することが必要である。しかしながら、情報システムを相互に結合するインターネットの超高速化に対応する不正パケット、不正攻撃検出システム（IDS）の実現は著しく困難であり、１０Gbps に対応する有効な IDS 方式は現在まで提案されていない。平木研究室では、今後数年間で予想される１０Gbps から４０Gbps のネットワーク速度領域におけるステートフルな IDS を実現するネットワークプロセッサアーキテクチャ研究を開始した。本研究では、超高速ネットワーク用 IDS に要請される処理速度を、FPGA を用いた再構成ハードウェアとソフトウェアベースのネットワークプロセッサに適応的分散により実現する。平成１５年度は研究の第一年度として、IDS の実現に必要な基本ハードウェアアルゴリズムの検討と、提案するネットワークプロセッサを実装するための２ポート・ネットワークインタフェースカードの試作を開始した。

本報告では、第２節において代用と再構成を用いた FPGA による超高信頼 CPU 研究内容をのべ、第３節で、超高速 IDS を実現するネットワークプロセッサ方式をしめし、第３節でまとめる。

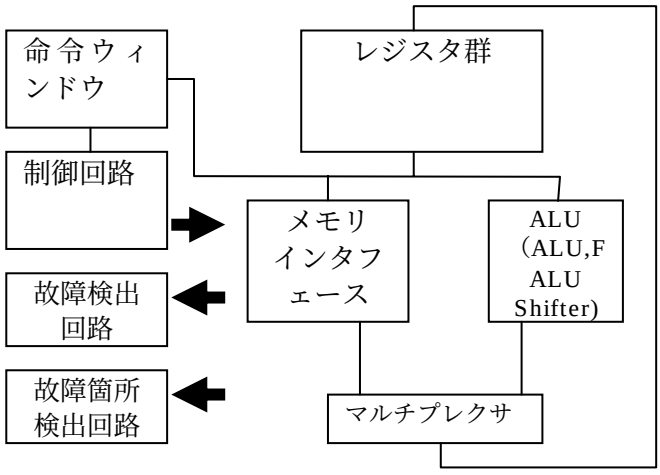
２．代用と再構成を用いる超高信頼 CPU

コンピュータは私たちの社会を支える根幹技術の一つであり、その重要性は年を経るごとに増しつつある。社会全体が計算システムに依存する現在、その故障により社会に深刻な被害を与える例も多々ある。非常に長期間の使用に耐え、完全にはシステムが停止しないコンピュータへの要求は増している。そのためには現代のコンピュータシステムを支える CPU、OS,ハードディスクなどの全ての部分要素における対故障性を再考慮する必要がある。本研究テーマでは計算の実行部分である CPU について議論する。CPU における故障は正常な命令実行を行わないという点で長期にわたり安定なシステムを構築する際の最大の障害であり、それに対するさまざまな手法が提案されてきた。回路の多重化、回路の代用は CPU 故障を回避する方法として、従来から用いられてきた。前者は同じ回路を複数所持し、壊れたときには残った正常動作をする回路を利用するという手法である。後者の手法はある部分の回路が壊れると別の回路を用いて代用を実現し、壊れた機能と同じ役割を手法である。前者の手法は回路サイズにおいて、後者は速度性能の大幅な低下が見られるという欠点を持つ。

これらの状況を踏まえ、私達は回路における再構成の手法と代用手法を組み合わせて用いることにより、回路に柔軟性を与え、故障においても速度性能の変化をもたらさないハードウェア再構成方式を提唱してきた。平成１５年度では、平成１４年度では実行部分に対する再構成手法を用いて故障回避を行ったことを拡張し、制御回路を含む CPU 全体に対する故障検出、回路の再構成と代用コード生成を行う CPU アーキテクチャを扱った。

２－１．基本アーキテクチャ

提案する CPU は、３２ビットの RISC アーキテクチャに準拠する命令セットを持つ。故障検出、故障回避の観点からみて、CPU は①演算器群、②命令制御回路（命令読み出し、命令発行、命令ウィンドウとレジスタ）および③制御回路と、それらの回路に対する故障検出回路、故障箇所検出回路で構成される。



故障検出は、制御回路では３重冗長化、命令制御回路（レジスタ群を含む）と演算器群では２重冗長化を用い、故障箇所検出は、FPGA 内部ロジックセルに対するバウンダリ・スキャンを仮定した。制御回路に３重冗長化を用いた理由は、①制御回路の論理回路量が、他の２つの部分と比較して著しく小さく、また、動作がハードウェア論理によっているため、代用を用いることがやや難しいためである。命令制御回路と演算器群は回路規模が大きいため、代用と再構成により故障修復を実現するため、２重冗長化回路による誤り検出だけを実装する。

２－２．再構成による配線遅延の変化

代用と再構成を用いる CPU の実現にとり、再構成に伴うクロック速度の低下（場合によっては向上）は大きな課題である。平成１５年度は、配線遅延の問題を明らかにするため、再構成に伴う配線遅延をシミュレーションにより求めた。

２－３．回路記述とシミュレーション

平成１４年度は、CPU 機能を C,C++で記述し、機能レベルのシミュレーションを行った。平成１５年度は、より正確で、論理回路に反映したシミュレーションを実現するため、全体を Sysetm　C（論理回路に変換することを用いた C 言語システム）で記述し、詳細シミュレーションを実施した。

３．超高速 IDS アーキテクチャ

IDS に必要な技術のうち、主に以下の 2 点が高速化研究の対象になっている。

（1）高速な文字列マッチ(exact match もしくは正規表現)
（2）TCP ストリームの認識(より一般に、トラフィック・ノーマライザ[1]

(1)は、IDS でパケットのペイロードに特定のパターンが含まれているかどうかをチェックする処理に使う。この部分は IDS の中でも特に処理が重い部分で、高速化の必要性が高い。有名なフリーの IDS ツールである Snort[2]を用いた研究が多い。
(2)は、安全性を高めたい場合は TCP パケットのストリームを組み立て直し、これに対してパターンマッチを適用する必要があるためである。さもないと、パケットを分割する等の手法により検査をすり抜けられてしまう。あまり網羅していないが、パケット順序交換等が無い場合について HW でサポートを行う方式として[3]や[4]がある。
他に、分割された IP パケットのデータ範囲が重なった場合のホストの振る舞いの曖昧性を利用して IDS をくぐり抜ける手法等がある。そこで、トラフィック・ノーマライザ[5]を使えば、セマンティックスに影響が無い範囲でパケットを作り直し、曖昧性を取り除く事ができる。曖昧性を取り除いた後の TCP ストリームに対して IDS を適用すればより高い安全性を確保できる。
しかし、このような処理を高帯域で行うためには、ハードウェアによる有効な方式を見つける必要がある。現在の所、実用に堪える方式はまだ無い。

今回提案する方式は（1）に含まれるが、（2）を意識したものである。具体的には、複数ストリームを扱う際のステートの退避・復帰が容易になるよう、ステートのサイズを減らしている。

３－１．文字列マッチアルゴリズム

IDS 実現で中心的役割をはたす文字列マッチは、従来以下のような方式で実現されてきた。

（1）Aho-Corasick アルゴリズム

IDS では複数の文字列パターンのマッチングを行う。このように複数の文字列を同時に対象文字列から探す古典的な(ソフトウェア向け)アルゴリズムとしては Aho-Corasick のアルゴリズム[5]がある。
このアルゴリズムでは、まず、探す文字列を全部含む TRIE を作る。次に、この TRIE 上の節点の位置をステートとするステートマシンを作り、各文字が来たときの状態遷移規則を作る。その後、探索対象文字列を 1 文字ずつこのステートマシンに入力し、文字列のマッチングを行う。
このアルゴリズムはステート遷移の際に表引きのためにメモリアクセスを行うため FPGA の実装では敬遠されているが、現在の自分の研究はこのアルゴリズムに基づいている。ソフトウェアによるアプローチでは、この方式を Snort に適用すると高速になる事を示した論文[6]等がある。

(2) NFA を用いた正規表現マッチ

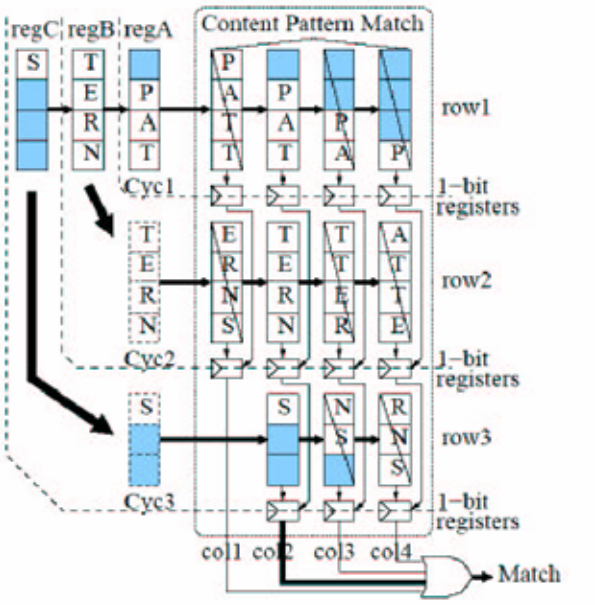
正規表現を受理する NFA を FPGA 上の回路で模倣して、文字列マッチを行う方式。マッチさせたい正規表現が与えられると、その正規表現でマッチを行う FPGA 上の回路を自動生成する。以下の FPGA による手法は、いずれもパターンが与えられると FPGA 上の回路を自動生成する。
NFAをシミュレートするために、NFAの各ノード毎にFFを一個用意し、そのノードがONかOFFかを表現する。長さnの正規表現を$O(n^2)$の回路量でサポートし、1 クロックあたり 1 文字ずつ入力文字列を処理できる。

一般の正規表現マッチアルゴリズムとしてこの方式を最初に提案したのは[7]で、IDS に最初に応用したのは[8]である。他に[4]や[9]等がこの方式を使用している。

この方式は正規表現をマッチできる点は強力だが、1 クロックあたり 1 文字しか処理できない点が苦しい。FPGA のクロック速度とネットワーク速度の差がどんどん開いているので、1 クロックで複数文字を処理できる事は必須である。
この方式のもう一つの問題点はステートの量が膨大 O(n)なため、ストリーム別マッチを行う場合にステートの退避・復旧が大変であるという点である。

(3) exact match を行う方式

IDS で行われるパケット内容検査のうち最も大きな割合を占めるのは exact match である[6]。[10]は、以下の図のような回路を使い 1 クロック毎に複数文字をマッチする。下の例では 1 クロックにつき 4 文字を処理する。この方式は 1 クロックで複数文字を処理できる点が優れているが、ステートのビット数はマッチする文字列中の文字の個数分となり、ステートの退避・復帰に問題が残る。Granidt[12]や[13]は FPGA 内に CAM を実装する事により文字列マッチを行う。この方式は CAM が大量の回路を使うため、小さな文字列セットしか扱う事ができない。また、CAM を大きくすると速度が落ちるという問題もある。



[14]は、Bloom Filter という一種のハッシュを使った方式である。マッチする文字列セットのハッシュ値をあらかじめ計算しておき、入力データのハッシュ値がそれと一致した場合はマッチする疑いがあるので、これを文字列マッチユニットに送り確認を行う。この方式では、マッチの可能性が無い場合は文字列マッチユニットが稼働しないようにする事で、処理速度が遅い文字列マッチユニットを用いてより高い帯域幅を実現する事を狙っている。欠点は、マッチする文字列の最大長が長くなると bloom filter が長くなってしまい、特に複数バイトを同時処理する時に厳しい点。それから、頻繁にマッチが起きる場合は文字列マッチユニットが全力で稼働せねばならない。

３－２．提案方式

私達は、文字列の exact match を行う方式として、以下の方式を提案する。

（1）1 クロックあたり複数バイトを同時に処理可能

（2）ストリーム毎のマッチを行う場合のステート退避・復旧が容易

これまで（1）を意識した研究はあるが、（2）を意識した先行研究はない。本研究では、（2）を実現するために前述の Aho-Corasick のアルゴリズムから出発した。このアルゴリズムで保持すべきステートは文字列 TRIE 中の節点番号のみで、マッチする文字列の合計長nに対して大体 O(log n)ビットのステートを保持すれば良い。ただし、この方式には以下の問題がある。

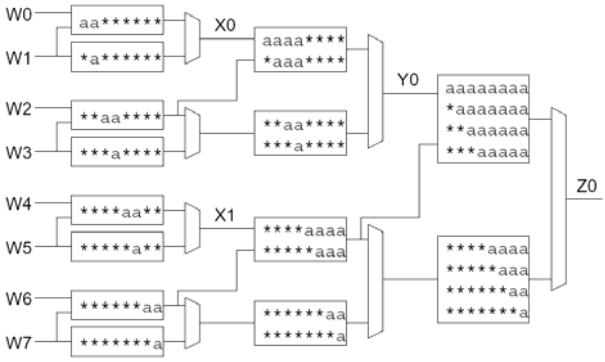
（1） そのままでは、複数バイトを一度に処理できない
（2）ステート遷移表を定数遅延で引くには、大きなメモリ(O(n^2))が必要

（2）は内部の高速 RAM が使えなくなるため速度が落ちる事に加え、後述の 1.の解決策を施した場合に多数(容量ではなくバンクが)の外部メモリが必要になるので問題である。

まず、（1）を解決するために複数バイト分同時にステートを進める方法を導入する。以下では、8 バイトを 1 クロックで処理する場合を考える。

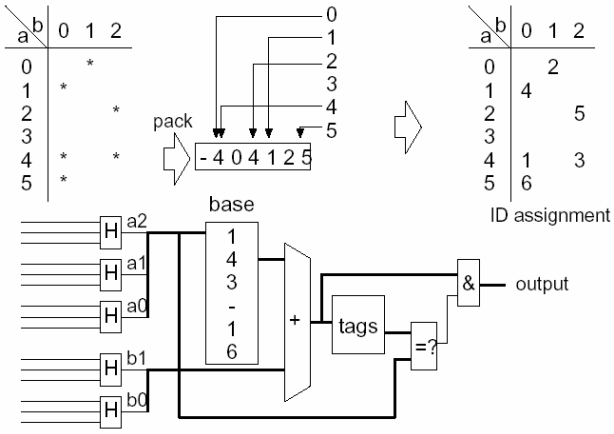
3-2-1. 複数バイトパターンの ID 付け

1 クロックで 8 バイト分ステートを進めるには、まず、各クロック毎に与えられた 8 バイトに対して、+aaaaaaaa+、+*aaaaaaa+、…、+*******a+、の 8 種類のパターンを認識する必要がある。ただし、+*+は don't care、+a+は特定の文字である。このパターンのうち、マッチすべき文字列の一部になっている物にはあらかじめ ID を付けてある。入力された 8 バイトについて上記パターンの longest match を行い、どれかのパターンに合致すれば、対応する ID を出力する。
以下の図がこれを実現する回路である。入力が 1 本の箱は、入力データに合致するパターンの ID(無ければ 0)を出力する。入力が 2 本の箱は、2 つの入力データを連結したものに合致するパターンの ID(無ければ 0)を出力する。入力 1 本の箱は単なる表引きなので説明しない。入力 2 本の箱の実装方法が肝である。全体をパイプライン化すれば、毎クロック 8 バイトの受付が可能である。



2 パターンの連結処理

入力 2 本の箱の中身は以下のようになっている。まず、2 つの入力に対してハッシュ値を計算する(各ビットは 3 入力 1 出力の論理関数。下図の"H")。ここで、複数の入力パターンが同じハッシュ値にならないようなハッシュ関数を使う。



ハッシュ値 a = (a2,a1,a0)と b = (b1, b0)の組み合わせのうち、有効なパターンが左上の表のようになっていたとする(+*+が有効な組み合わせ)。この 2 次元の表の各列を+*+

の部分が重ならないように 1 次元の配列にパックし(中央上)、各 a の値毎に対応する列の開始位置ポインタを記録する。1 次元の配列の各要素には、これが何番目の列の物か判別できるようにするために列番号を格納する。1 次元配列の左から数えた位置が、パターンの ID となる。

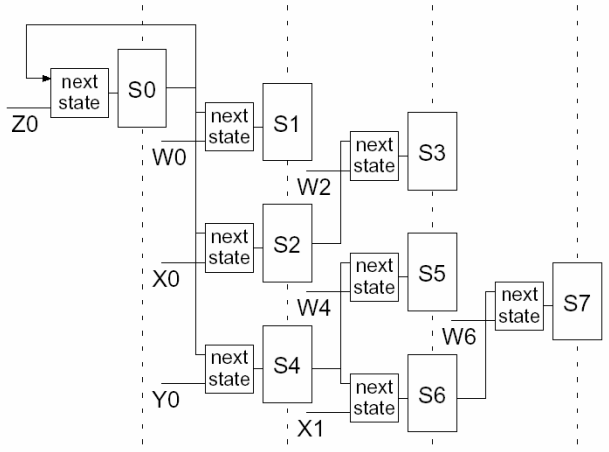
与えられた入力の ID を求める回路は以下のようになる。base が各列の開始位置ポインタ表で、tags が列番号がパックされた 1 次元配列である。

この方式では、ハッシュ関数をうまく作ってできるだけ bのビット数を減らす事が重要である。現在は近似アルゴリズムの動作を試験中である。ハッシュの各ビットの論理関数が 3 ビットに制限されているのはハッシュ関数最適化ルーチンの計算量の制約のため。また、1 次元配列へのパックもできるだけ隙間が生じないようにする必要があるが、これは長い列から順に欲張り法で格納する予定である。

ステートの計算

最後に、ステートを求める部分は以下のようになる。S0 は 1 クロック毎に 8 バイト進むステートで、入力文字列の 8 バイト境界におけるステートを保持する。このままだと 8 バイト境界以外で終了するパターンを発見できないので、先の 8 バイトパターン ID 付け回路の W0、W2、W4、W6、X0、X2、Y0 の信号(をシフトレジスタで適切なサイクル分遅らせた物)を用いてそれ以外のバイト境界のステートを順次計算する。

next state を計算する場合も、パターン ID 付けと同様に 2 次元表引きが必要となるが、これも前述の方式と同様に実装する。なお、S0 の計算だけは 1 クロックで計算する必要があるので、ここだけは物量を投入してクロック遅延を減らす。



ストリーム毎のステート退避・復旧

先の図の S0 のみを退避・復旧すれば良い。S1--S7 は S0 に追従して変化するだけなので、ストリームを切り替えれば自動的に追従する。

4．おわりに

平木研究室で行った 2 個の研究テーマは、異なるアプローチで情報システムの超高信頼性を実現することを目的としたものである。代用と再構成を用いる超高信頼 CPU アーキテクチャの研究では、今後詳細なシミュレーションを実施するとともに、実際の FPGA 上に提案アーキテクチャを実装し、評価する予定である。

また、超高速 IDS アーキテクチャの研究では、暗号化通信による、パケット解析の困難さを解決し、有効な IDS を構築する予定である。そのような場面で使う方式としては以下のようなものが考えられる。

（1）エッジルータで通信の暗号化をまとめて行い、ここに IDS を組み込む(同時にトラフィック・ノーマライザも組み込む)

（2）ホストベース方式で、NIC が TCP 処理、暗号化、IDS を同時に行う。

参考文献

[1] Mark Handley, Vern Paxson, “Network Intrusion Detection : Evasion, Tra_c Normalization, and End-to-End Protocol Semantics," in Proc. of 10th USENIX Security Symposium, August 2001.

[2] http://www.snort.org.

[3] Marc Necker, Didier Contis, David Schimmel, “Tcp-stream reassembly and state tracking in hardware," in Proc. of 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02), pp. 286-287, September 2002.

[4] David V. Schuehler, James Moscola, John Lockwood, “Architecture for a Hardware Based, TCP/IP Content Scanning System," in Proc. of 11th IEEE Symposium on High Performance Interconnects(HotI '03), pp. 89-94, August 2003.

[5] A. V. Aho, M. J. Corasick,”cient String Matching : An Aid to Bibliographic Search," Communications of the ACM, vol. Vol. 18, pp. 333 - 340, June 1975.

[6] C. J. Coit, S. Staniford, J. McAlerney, “Towards Faster String Matching for Intrusion Detection or Exceeding the Speed of Snort," in DISCEXII, DARPA Information Survivability conference and Exposition, 2001.

[7] R. Sidhu, V. K. Prasanna, “Fast regular expression matching using fpgas," in Proc. of 9th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01), April - May 2001.

[8] B. L. Hutchings, R. Franklin, D. Carver, “Assisting network intrusion detection with recon_gurable hardware," in Proc. of 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02), pp. 111-120, September 2002.

[9] James Moscola, John Lockwood, Ronald P. Loui, Michael Pachos, “Implementation of a content scanning module for an internet _rewall," in Proc. of 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'03), pp. 31 - 38, April 2003.

[10] Y. H. Cho, S. Navab, W. H. Mangione-Smith, “Specialized Hardware for Deep Network Packet Filtering," in Proc. of 12th International Conference on Field Programmable Logic and Applications(FPL'02), September 2002.

[11] Ioannis Sourdis, Dionisios Pnevmatikatos, “Fast, Large-Scale String Match for a 10Gbps FPGA-based Network Intrusion Detection System," in Proc. of 13th International Conference on FieldProgrammable Logic and Applications(FPL '03), September 2003.

[12] Maya Gokhale, Dave Dubois, Andy Dubois, Mike Boorman, Steve Poole, Vic Hogsett, “Granidt:Towards Gigabit Rate Network Intrusion Detection Technology," in Proc. of 12th International Conference on Field Programmable Logic and Applications(FPL '02), September 2002.

[13] Shaomeng Li, Jim Torresen, Oddvar Soraasen, “Exploiting recon_gurable hardware for network security," in Proc. of 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'03), pp. 292-293, April 2003.

[14] Sarang Dharmapurikar, Praveen Krishnamurthy, Todd Sproull, John Lockwood, “Deep packet inspection using parallel bloom _lters," in Proc. of 11th IEEE Symposium on High Performance Interconnects (HotI '03), pp. 44 -51, August 2003.