5K-5

SmartCore システムのデッドロック回避

笹河 良介 †　佐藤 真平 † ‡　吉瀬 謙二 †
東京工業大学大学院 情報理工学研究科 † 日本学術振興会 特別研究員（DC2）‡

1 はじめに

SmartCore システムは，メニーコアプロセッサおいて複数のノードを用いた冗長実行により信頼性向上などを達成するシステムである [1]．ネットワークにおいて，次に入るべきチャネルの資源が解放されず，今入っているチャネルの資源を永遠に確保してしまうことによって起こる**デッドロック**．このデッドロックが，通信毎にチャネルを分けた場合の SmartCore システムでは回避出来ていることを証明する．

2 SmartCore システムのデッドロック回避

2.1 デッドロック回避の証明のための定義および補題

定義 1. *Interconnection network* I は $I = G(N, C)$ で表される強連結な有向多重グラフである．グラフの頂点 $n_i \in N$ は各ノード，グラフの辺 $c_i \in C$ は各ノードを繋ぐチャネルとなる．仮想チャネルはそれぞれ区別し，各チャネル c_i は，辺の始点となるノード s_i から辺の終点となるノード d_i へのフリットの通信が可能である．

定義 2. *Routing function* $\mathbf{R} : C \times N \to \mathcal{P}(C)$ ($\mathcal{P}(C)$ は C のべき集合) は，現在のチャネル c_i と宛先となるノード n_d を受け取り，ルーティングアルゴリズムに則って転送先となりうるチャネルの集合を返す関数である．各チャネルは自身へルーティングされない．つまり $c_j \in \mathbf{R}(c_i, n_d) \Rightarrow c_i \neq c_j$.

定義 3. *Channel dependency graph* D は，Interconnection network I と Routing function $\mathbf{R}$ が与えられたときに，$D = G(C, E)$ で表される有向グラフである．グラフの頂点 c_i はチャネル，グラフの辺は $\mathbf{R}$ によって決められる以下のチャネルの組の集合である．

$$E = \{(c_i, c_j) \mid c_j \in \mathbf{R}(c_i, n) \text{ for some } n \in N\}$$

Channel dependency graph の辺は Routing function によって決定されるので，「辺の終点となるチャネルの資源が解放されないと辺の始点となるチャネルの資源が解放されない」ことを意味している．

定義 4. チャネル c_i において，バッファに格納されているパケットの宛先となるノードが n_d であるとき，$head(c_i) = n_d$ と書く．

定理 1. Interconnection network I と Routing function $\mathbf{R}$ から生成される有向グラフ Channel dependency graph D において閉路が無いならば，その I における $\mathbf{R}$ はデッドロックフリーである．

2.2 デッドロック回避の証明

2 重実行時における SmartCore システムは，通常のノード間通信に加え，パケット比較のためにミラーノードがマスターノードへ通信するマージ通信，他ノードから届いたパケットをマスターノードがミラーノードへ通信するコピー通信が行われる．また，マージ通信（ノード間通信）において，その通信のパケットが宛先となるノードに到達した際，対応するノード間通信（コピー通信）のパケットが利用するチャネルの資源が確保出来るまで，マージ通信（ノード間通信）のパケットはチャネルを解放されない．つまり，マージ通信（ノード間通信）のパケットがネットワークから排出されるチャネルから，対応するノード間通信（コピー通信）のパケットが注入されるチャネルへ，新しい Channel dependency graph の辺ができる．この新しい Channel dependency graph の辺がどのように定義出来るかを以下に示す．

定義 5. *Injection routing function* $\mathbf{R}_{inject} : N \times N \to \mathcal{P}(C)$ は，送り元となるノードと宛先となるノードを受け取り，ルーティングアルゴリズムに則って，最初にパケットが注入されうるチャネルの集合を返す関数である．

定義 6. *Ejection routing function* $\mathbf{R}_{eject} : N \times N \to \mathcal{P}(C)$ は，送り元となるノードと宛先となるノードを受け取り，ルーティングアルゴリズムに則って，宛先となるノードに着いて最終的にパケットが排出される際のチャネルの集合を返す関数である．

定義 7. $N_m \subseteq N$ は，マスターノードの集合．対応するスレイブノードを持たないノードもマスターノードとする．

定義 8. $slave : N \to N$ は，ノードを受け取り，対応するスレイブノードを返す関数．対応するスレイブノードが存在しない場合は，$\emptyset$ を返すものとする．

定義 9. E_{pv} はパケット比較によって，マージ通信のパケットがネットワークから排出されるチャネルから，対応するノード間通信のパケットが注入されるチャネルへできる，新しい Channel dependency graph の辺である．以下のように定義する．

$$E_{pv} = \{(c_i, c_j) \mid c_i \in \mathbf{R}_{eject}(slave(n), n),\ c_j \in \mathbf{R}_{inject}(n, m) \text{ for } n \in N_m, m \in N_m, slave(n) \neq \emptyset\}$$

定義 10. E_{pu} はパケット複製によって，ノード間通信のパケットがネットワークから排出されるチャネルから，対応するコピー通信のパケットが注入されるチャネルへできる，新しい Channel dependency graph の辺である．以下のように定義する．

$$E_{pu} = \{(c_i, c_j) \mid c_i \in \mathbf{R}_{eject}(m, n),\ c_j \in \mathbf{R}_{inject}(n, slave(n)) \text{ for } n \in N_m, m \in N_m, slave(n) \neq \emptyset\}$$

定義 11. D_{smart} は SmartCore システムにおける Channel dependency graph であり，$D_{smart} = G(C, E_{smart})$ 表される．$E_{smart} = E \cup E_{pv} \cup E_{pu}$.

さらにここで，1 つ定理を示す．

定理 2. グラフ D_{smart} において閉路が無いとき，その SmartCore システムはデッドロックフリーである．

証明： グラフ D_{smart} において閉路が無いとき，D_{smart} は有向非巡回グラフ（Directied Acyclic Graph, DAG）である．したがって，グラフのノード C に対してトポロジカルソートが施せる．ここで，順序付けされたチャネルに対して先頭から，0 から $|C| - 1$ までのラベルを付ける．各チャネルは $label(c_i)$ で自身に付けられたラベルを返すとする．

Deadlock avoidance on SmartCore system
Ryosuke SASAKAWA †, Shimpei SATO † ‡, and Kenji KISE †

† Graduate School of Information Science and Engineering, Tokyo Institute of Technology
‡ Research Fellow of the Japan Society of the Promotion of Sciensce (DC2)

$label(c_i) =\mid C \mid -1$ **の時**
チャネル c_i から伸びる Channel dependency graph の辺は無い．したがって，チャネル c_i に入ったパケットは他のチャネルに関係なくネットワークから排出される．よって，チャネル c_i に入ったパケットは入力バッファから出て行き，チャネル c_i の資源は解放される．

$label(c_i) = k(0 \leq k <\mid C \mid -1)$ **の時**
$k < label(c)$ となるチャネル c 全ての資源が解放されると仮定する．チャネル c_i から伸びる Channel dependency graph の辺が存在しない場合，$label(c_i) =\mid C \mid -1$ の時と同様に，チャネル c_i の資源は解放される．チャネル c_i から伸びる Channel dependency graph の辺が存在する場合，$head(c_i) = d_i$ および $head(c_i) \neq d_i$ で場合分けができる．

$head(c_i) = d_i$ **のパケットが入っている時**
E_{pv} および E_{pu} の辺が存在しない場合，c_i に入っているパケットは，他のチャネルに関係なくネットワークから排出される．E_{pv} または E_{pu} の辺が存在する場合，該当する E_{pv} または E_{pu} の辺の終点となるチャネルは $k < label(c)$ となるチャネルであり，$k < label(c)$ となるチャネルの資源はは全て解放される．したがって，チャネル c_i に入っているパケットに対応するノード間通信またはコピー通信のパケットは資源を確保することができ，チャネル c_i 入っているパケットはネットワークから排出される．よって両方の場合においてチャネル c_i の資源は解放される．

$head(c_i) \neq d_i$ **のパケットが入っている時**
SmartCore システムにおいても全てのノード同士に通信経路が存在し，Interconnection network I は強連結なグラフである．よって，$\mathbf{R}(c_i, head(c_i)) \neq \emptyset$ であり，E の辺が存在する．該当する E の辺の終点となるチャネルは $k < label(c)$ となるチャネルであり，$k < label(c)$ となるチャネルの資源はは全て解放される．したがって，チャネル c_i に入っているパケットは $\mathbf{R}$ によって選ばれたチャネルの資源を確保することができる．よって，チャネル c_i 入っているパケットはチャネル c_i の入力バッファから出ることができ，チャネル c_i の資源は解放される．

したがって，$label(c_i) = k(0 \leq k <\mid C \mid -1)$ の時もチャネル c_i に入ったパケットは入力バッファから出て行き，チャネル c_i の資源は解放される．

以上より，帰納的に全てのチャネルに入ったパケットは入力バッファから出て行き，チャネルの資源が解放されることが示された．よって，デッドロックフリーとなる．したがって，グラフ D_{smart} において閉路が無いとき，その SmartCore システムはデッドロックフリーであることが示された． □

定理 3. マージ通信用チャネル，ノード間通信用チャネル，コピー通信用チャネルをそれぞれ用意した場合の SmartCore システムはデッドロックフリーである．

証明： マージ通信用チャネルの集合を C_{merg}，ノード間通信用チャネルの集合を C_{node}，コピー通信用チャネルの集合を C_{copy} とする．すると全チャネルの集合は $C = C_{merg} \cup C_{node} \cup C_{copy}$ とできる．

$I = G(N, C)$ における $D_{smart} = G(C, E_{smart})$ に閉路が無いことを示す．

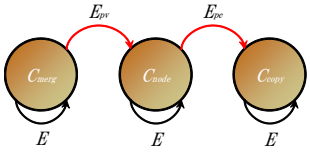

図 1: 各チャネルに入ったフリットの行き先

SmartCore システムにおいては以下のことが成り立つ．

$$\begin{cases} c_i \in C_{merg} \wedge head(c_i) \neq d_i \Rightarrow c_j \in \mathbf{R}(c_i, head(c_i)), c_j \in C_{merg} \\ c_i \in C_{merg} \wedge head(c_i) = d_i \Rightarrow \mathbf{R}(c_i, head(c_i)) = \emptyset \\ c_i \in C_{node} \wedge head(c_i) \neq d_i \Rightarrow c_j \in \mathbf{R}(c_i, head(c_i)), c_j \in C_{node} \\ c_i \in C_{node} \wedge head(c_i) = d_i \Rightarrow \mathbf{R}(c_i, head(c_i)) = \emptyset \\ c_i \in C_{copy} \wedge head(c_i) \neq d_i \Rightarrow c_j \in \mathbf{R}(c_i, head(c_i)), c_j \in C_{copy} \\ c_i \in C_{copy} \wedge head(c_i) = d_i \Rightarrow \mathbf{R}(c_i, head(c_i)) = \emptyset \end{cases}$$

また，定義 9 および 10 より以下が成り立つ．

$$\begin{aligned} (c_i, c_j) \in E_{pv} &\Rightarrow c_i \in C_{merg} \wedge c_j \in C_{node} \\ (c_i, c_j) \in E_{pu} &\Rightarrow c_i \in C_{node} \wedge c_j \in C_{copy} \end{aligned} \quad (1)$$

つまり，

$$\begin{cases} c_i \in C_{merg}, (c-i, c_j) \in E_{smart} \Rightarrow c_j \in C_{merg} \vee c_j \in C_{node} \\ c_i \in C_{node}, (c-i, c_j) \in E_{smart} \Rightarrow c_j \in C_{node} \vee c_j \in C_{copy} \\ c_i \in C_pcopy, (c-i, c_j) \in E_{smart} \Rightarrow c_j \in C_{copy} \end{cases}$$

以上から，Channel dependency graph の辺 (c_i, c_j) が，各集合に属するチャネルから，どの集合に属するチャネルに伸びているのかを表したグラフを図 1 に示す．

Channel dependency graph D_{smart} が閉路を持つためには，

（C_{merg} に属するチャネルから C_{merg} に属するチャネルへ伸びる辺のみで構成される閉路がある）
$\vee$（C_{node} に属するチャネルから C_{node} に属するチャネルへ伸びる辺のみで構成される閉路がある）
$\vee$（C_{copy} に属するチャネルから C_{copy} に属するチャネルへ伸びる辺のみで構成される閉路がある）

が真である必要がある．

ここで，「C_{merg} に属するチャネルから C_{merg} に属するチャネルへ伸びる辺」は全て XY 次元順ルーティングによって形成された Channel dependency graph の辺であるので，補題から「C_{merg} に属するチャネルから C_{merg} に属するチャネルへ伸びる辺のみで構成される閉路がある」は成り立たない．同様のことが C_{node} および C_{copy} にも言える．したがって上記の命題は成り立たず，Channel dependency graph D_{smart} は閉路を持たない．よって，定理 2 より Channel dependency graph が閉路を持たないならばデッドロックフリーであるので，マージ通信用チャネル，ノード間通信用チャネル，コピー通信用チャネルをそれぞれ用意した場合の SmartCore システムはデッドロックフリーであることが示された．

□

謝辞
本研究の一部は，JST CREST 「アーキテクチャと形式的検証の協調による超ディペンダブル VLSI」の支援による．

参考文献

[1] 池田貴一, 佐藤真平, 吉瀬謙二. 冗長実行時の SmartCore システムの性能評価. 情報処理学会研究報告 2011-ARC-197, No. 32, 2011.