Simulinkモデルからのブロックレベル並列化

山口 滉平[1]　竹松 慎弥[1]　池田 良裕[1]　李 瑞徳[1]　鍾 兆前[1]　近藤 真己[2]　枝廣 正人[1,a)]

概要：本発表では，Simulink モデルにおけるブロック構造を抽出し，並列化を行う設計フローを示す．ここではブロックレベル構造をあらわす XML 記述を策定し，はじめに Simulink モデルから変換する．これに対し，自動生成コードから抽出したコード列をブロック単位で付加し，ハードウェア抽象化記述である SHIM を用いて粒度見積を行う．その上で，階層クラスタリング法を用いたグラフ分割アルゴリズムを用いてコア割当を行い，コード列とコア割当情報を用いて並列化コード生成を行う．

キーワード：マルチコア，組込み制御，並列化，モデルベース開発，Simulink

1. はじめに

電力制約の厳しい組込みシステムでは，システムの高性能化に伴い，プロセッサの動作周波数向上ではなく，マルチ・メニーコア化により性能向上を達成することが必須である．しかしながら，特に組込み制御分野においては，従来より逐次処理の多い設計が主流で，Cコードレベルでの並列化が難しいことが知られている．一方で，車載制御機器などでは，設計の上流シフトが進んでおり，Mathworks 社の MATLAB/Simulink [1] に代表されるようなモデルベース開発ツールを用いた設計が進んでいる．

このような背景のもと，我々は，Simulink モデルからの並列化ツールを研究し，モデルベース開発において設計者が並列動作を考慮することができ，マルチ・メニーコア向け実装を支援するような設計環境を確立することを目指している．本発表では，我々が構築している設計フローの概要について紹介する．

2. 準備

2.1 Simulink モデル

Simulink モデルはブロック線図を用いてシステムの動作をあらわす．各ブロックは加算，ゲインといった基本演算から，微積分，PID 制御といったように内部状態を有する複雑な演算まで多様な種類があり，また，Data Store とよばれるようなメモリ機能を持つブロックもある．

これらのブロックを線でつないでシステムの動作を表現し，シミュレーションが可能である．また，特殊なブロックを除き，Cコード自動生成も可能であり，実装可能な仕様としての方向性も持つ．本発表ではコード生成可能な Simulink モデルを対象とする．

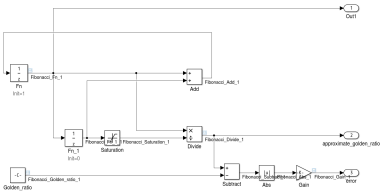

図 1　Simulink モデルの例
Fig. 1　Example of Simulink Model.

現状ではシミュレーションを並列動作するための API 群は有しているが，自動生成される実装コードは逐次的である．Simulink モデルの例を**図 1** に示す．

2.2 SHIM

SHIM (Software-Hardware Interface for Multi-many-core) [2] は，国際標準化団体 Multicore Association において標準化された，ソフトウェア，ツールのためのハードウェア抽象化記述である．

SHIM においては，コア，メモリ，アドレスマップなどの構成情報が XML 記述を用いて記載されている他，各コアに対し，LLVM 中間言語レベル [3] での遅延値，各コアから各アドレスに参照した際の遅延値が，それぞれ best, typical, worst 値として表現されている．

2.3 BLXML

BLXML (Block-Level structure XML) は，組込みマルチコアコンソーシアム [4] において検討中の構造で，ブロック線図構造を持つ，モデルと実装を相互に参照し，協調設計を可能とするためのデータ構造である．ブロック線図構造の他，各ブロックに対応するCコード，それらから計測

[1] 名古屋大学
Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464–8603, Japan
[2] NEC 情報システムズ
NEC Informatec Systems, Shimonumabe, Nakahara-ku, Kawasaki 211–8666, Japan
[a)] eda@ertl.jp

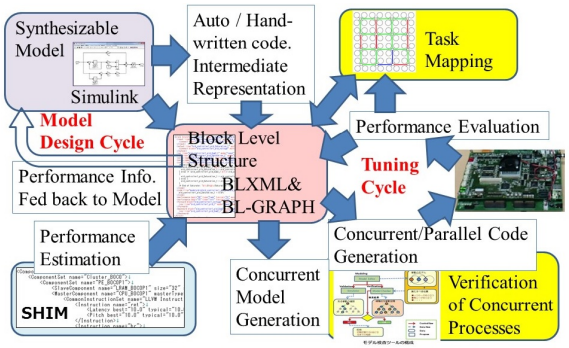

図 2　並列化設計フロー
Fig. 2　Parallelization Design Flow.

した粒度情報（ブロック単位の処理量情報），並列化後のコア配置情報も持つことができる．

3.　並列化設計フロー

図 **2** は我々の並列化設計フローである．

3.1　BLXML 生成

3.1.1　ブロックレベル構造取得

Simulink からブロックレベル構造を抽出し，BLXML を生成する．Simulink は多くの分野に適合するために広い仕様が定義されているが，ここでは組込み制御で標準的に用いられる仕様を含む範囲として，QGen [5] が対応するブロックに標準的に対応している．

3.1.2　コード付加

各ブロックに対応するコードを付加する．Simulink モデルから生成されたコードは，ほぼブロックと対応が取れ，その情報がコードのコメント等に記載されることを利用し，対応を取る．ただし，複数のブロックから一括してコード生成される場合もあり，それらの情報についても付加する．

3.1.3　粒度見積

各ブロックに対応するコードから粒度（処理量）を見積る．具体的には，Clang 等 LLVM コンパイラにより LLVM 中間言語列に展開し，SHIM を用いて見積る [6]．

3.2　並列化

ブロックのグラフ構造と粒度を用い，並列化を行う．ただし，多くの場合，組込み制御特有の処理開始ブロック，終了ブロックが存在し，かつ，それぞれの開始，終了に対応する制御周期が存在する．そのため，基本的にグラフ構造をコア数に分割する階層クラスタリング手法 [7] を用いるが，クリティカルパス解析 [8] を併用している．

コア配置結果は BLXML に書き戻され，それをもとに Simulink モデルにフィードバックすることも可能である．現在はコア配置情報により，ブロックの色分け表示を行っている．図 1 に対して 4 コア向け並列化を行った結果を図 **3** に示す．

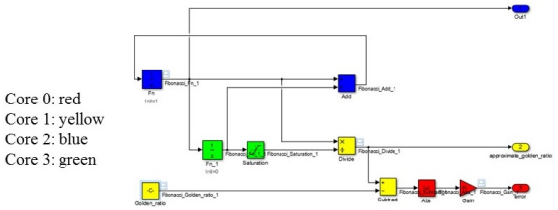

図 **3**　図 1 に対する並列化結果
Fig. 3　Parallelization Results for Fig. 1.

3.3　並列化コード生成

各ブロックの配置情報，周期情報，コード断片情報より並列化コードを生成する．コード生成部はリターゲッタブルになっており，現在のところ，pthread, XC (XMOS) [9], SigmaC (KALRAY) [10]，eMCOS スレッド [11] 向けなどを生成可能である [12], [13]．

また，生成されたコードは，Simulink 向け自動テスト生成ツール Reactis [14] により作成されたテストケースにより，自動的にテストされる．

4.　まとめと今後の課題

Simulink からのブロックレベル並列化設計フローについて紹介した．今後は，Simulink 仕様拡大，並列性能向上を進めるとともに，検証および SHIM を用いた見積精度向上が課題である．

謝辞　いつもご指導いただく，組込みマルチコアコンソーシアム会員および共同研究各社の皆様，名古屋大学道木研究室の皆様に深く感謝します．

参考文献

[1] Mathworks: http://www.mathworks.com/products/simulink/.
[2] Multicore Association: http://www.multicore-association.org/workgroup/shim.php.
[3] LLVM: http://llvm.org/docs/LangRef.html.
[4] 組込みマルチコアコンソーシアム: http://www.embeddedmulticore.org/.
[5] AdaCore: http://www.adacore.com/qgen.
[6] 西村，中村，荒川，枝廣: ソフトウェア向けハードウェア性能記述を用いたマルチコアにおける性能見積り, 情報処理学会研究報告, 2014-EMB-32, 26, 2014.
[7] 油谷，枝廣: 階層構造を持つメニーコアアーキテクチャへのタスクマッピング, 情報処理学会論文誌, Vol.56 (2015), No. 8, pp.1568 - 1581.
[8] 山田，枝廣: モデル予測制御における非線形漸化式実行の並列化, 情報処理学会研究報告, 2015-EMB-36, 37, 2015.
[9] XMOS: http://www.xmos.com/.
[10] KALRAY: http://www.kalrayinc.com/.
[11] eSOL: https://www.esol.co.jp/embedded/emcos.html.
[12] 大川，枝廣: マルチレート制御モデルのイベントドリブンプロセッサ実装, 情報処理学会研究報告, 2013-EMB-30, 2, 2013.
[13] R. Nakamura, F. Arakawa, and M. Edahiro: “Simple One-to-one Architecture for Parallel Execution of Embedded Control Systems,” *CPSNA’14*, pp.25-30, 2014.
[14] Reactive Systems: http://www.reactive-systems.com/.