メニーコアプロセッサにおける実時間モデル予測制御のための投機実行法

川 上 哲 志[†] 岩 永 明 人[††] 井 上 弘 士[†††]

本稿では，メニーコアプロセッサによる実時間モデル予測制御向けの投機実行法を提案する．提案手法では，モデル予測制御が特定の処理である最適制御問題を周期的に計算することに着目し，最適制御問題をメニーコアプロセッサで投機実行することでリアルタイム処理の実現を狙う．本稿では，投機実行に必要な状態値の予測手法を提案し，予測ヒット率を求めることで提案手法の有効性を評価した．振り子制御を例とした計算機実験により，提案手法は状態値の予測が可能であることを確認した．

Speculative Execution for Real-time Model Predictive Control on Manycore Processor

SATOSHI KAWAKAMI,[†] AKIHITO IWANAGA[††] and KOJI INOUE[†††]

This paper presents a speculative execution for real-time model predictive control on many-core processor. Our method focuses on periodic calculation of optimal problems ,and speculatively executes optimal control problems with many-core processor in real time. We propose a method for predicting the state values required for the speculative execution, and evaluated hit rates for the number of processor cores. Our experimental results show that our method predict the state value for a pendulum control.

1. はじめに

制御工学の進歩は，航空機，自動車，船舶などの様々な製造業において，製品の精度の改善や高性能化に大きく寄与してきた．さらに，高層ビルや橋梁の建設などの製造プロセスにおける生産性の向上にも重要な役割を果たしている．次世代の制御システム開発においては，計算量が大きく，要求されるレイテンシが小さい処理を実現する計算機システムが求められている．

近年，モデル予測制御（MPC:Model Predictive Control）と呼ばれる制御手法が注目されている．MPC は，制御対象のモデルを利用してシステムの状態値を予測し，操作量を求める制御手法である．制御対象の動的モデルを正確に定式化することで，将来のシステムの状態値の変化を予測し，状態値と目標値が一致する操作量を決定することが可能になる．しかしながら，MPC は従来の制御手法と比較して，サンプリング周期毎に計算量の大きい最適制御問題を解く必要があるため，MPC のリアルタイム処理が課題である[1)]．

一方，半導体の微細化技術の発展により，制御アプリケーションを実行するマイクロプロセッサは順調に性能向上を実現してきた．特に，動作周波数向上の限界に直面した 2000 年初頭以降，1 つのチップに複数のプロセッサコアを搭載するマルチコアが主流となっており，今後は 64〜128 個のコアを搭載したメニーコアへと発展していくことが予想される[2),3)]．このようなオンチップ並列処理においては，如何に並列性を抽出できるかが性能向上の最重要課題となる．しかしながら，MPC では，1) 小さなサイズの入力データが時系列に入力され，かつ，2)MPC の数値計算は逐次処理が支配的である，といった性質を持つ．すなわち，従来のスレッド／データレベルの空間並列処理を適用することは本質的に困難であり，メニーコアの潜在能力を活かすことができない問題が生じる．

そこで本稿では，MPC の適用範囲をより厳しいリアルタイム処理応用へと拡大すべく，MPC の高速化を目的とした新しいメニーコア活用法を提案する．具体的には，MPC が特定の処理である最適制御問題を周期的に計算する点に着目し，最適制御問題を複数のプロセッサコアで投機的に実行することで，MPC の実時間処理を狙う．提案手法では，投機実行で予測した状態値を用いて最適制御問題を解くことで MPC を高速化する．本稿では，投機実行に必要な状態値の予測手

† 九州大学大学院システム情報科学府情報知能工学専攻
Dept. of Advanced Information Technology, Kyushu Univ., kawakami@soc.ait.kyushu-u.ac.jp
†† 九州大学大学院統合新領域学府オートモーティブサイエンス専攻
Dept. of Automotive Science, Graduate School of Integrated Frontier Sciences, Kyushu Univ.
††† 九州大学大学院システム情報科学研究院情報知能工学部門
Dept. of Advanced Information Technology, Kyushu Univ.

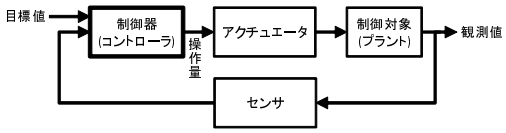

図 1 フィードバック制御のブロック図
Fig. 1 Block diagram of feedback control systems.

法を提案し，予測値の近傍を含めた投機実行の性能を予測ヒット率の導入によりモデル化する．予測ヒット率を定量化することで，提案手法の有用性を議論する．

本論文の構成は以下の通りである．2 節では，制御の概要，今後の制御システムと課題，及びモデル予測制御について述べる．3 節では，提案手法である最適制御問題の投機実行の概要と状態値の予測手法について述べる．4 節では，予測値と観測値の差異を実験的に示し，投機実行の有効性について議論する．5 節では，MPC プログラムの高速実行の関連研究をまとめる．最後に，6 節で本論文をまとめる．

2. 制御システムの傾向とモデル予測制御

2.1 制御システムの分類

制御とは，制御対象を意図した通りに操作することである．図 1 に一般的なフィードバック制御を示す．制御器は，制御対象からセンサを介して得た観測値に基づき，操作量を決定する．制御の手法として，MAP 制御，PID 制御，及び MPC が挙げられる．

MAP 制御では，予め観測値と目標値に対する操作量を実測，またはモデルに基づき計算し，観測値に対応する操作量をルックアップテーブルで保持する．稼働時は，ルックアップテーブルを参照することで操作量を決定する．MAP 制御は，自動車のエンジン制御といった多入力なシステムで用いられている．

PID 制御とは，操作量を観測値と目標値の偏差・積分・微分の 3 要素によって決定する制御手法である．PID 制御は，一般的に SISO (Single Input Single Output) のシステムで用いられ，実装の容易さから最も幅広く利用される．

MPC とは，制御対象から制御に関連する特性を数学的モデルとして定式化し，モデルに基づいてシステムの挙動を予測することで操作量を決定する．MPC は，主に化学プラントで利用されている．

2.2 今後の制御システムと課題

今後の制御システムで必要とされる計算機環境を議論するために，要求されるレイテンシと制御システムの複雑度の観点から，各制御手法の特徴をまとめる．現在，要求レイテンシが小さく，複雑度が低い，モータ，エンジン，及びロボットアーム制御などに対しては，MAP 及び PID 制御が採用されている．一方，要求レイテンシが大きく，複雑度が高い，化学プラント，及び発電所などに対しては，MPC が採用されている．

次世代の自動車制御やロボット制御システムにおいては，要求レイテンシが小さく複雑度の高いシステムの制御を実現する制御手法が必要であると予測される．次世代の制御システムの実現に向けては，MAP/PID と，MPC を用いた 2 つのアプローチが考えられる．MAP 及び PID 制御を用いたアプローチでは，テーブルサイズの肥大化，入出力数の制約から，複雑度の高い制御システムへの適用は困難である．一方，MPC ではシステムの振る舞いを定式化すれば，様々なシステムの制御が可能である．特に，一般的な制御系設計手法が存在しない非線形システムにおいて，MPC は有効な制御手法である．MPC では，処理手法を改善することにより，要求レイテンシの小さいシステムへの適用を可能にする研究が行われている [4]．本稿では，MPC を用いたアプローチに着目し，メニーコアプロセッサによるリアルタイム MPC が可能な実行法の構築を狙う．

2.3 モデル予測制御（Model Predictive Control）

MPC とは，制御対象のモデルを利用してシステムの状態値を予測し，操作量を求める制御手法である．制御対象の動的モデルを正確に定式化することで，制御システムの過渡及び定常状態の振る舞いを正確に制御することが可能である．MPC はサンプリング周期毎に有限区間の最適制御問題を解くことで操作量を決定する．MPC は一般的に非線形システムを対象とし，非線形システムは，

$$\dot{x}(t) = f(x(t), u(t), t) \tag{1}$$

と表される．ただし，$x(t) \in \mathbb{R}^n$ は状態ベクトルで，$u(t) \in \mathbb{R}^m$ は操作量のベクトルである．式 (1) に対して，最適制御問題は以下のように定式化される．

minimize
$\phi(x^*(t+T), t+T) + \int_t^{t+T} L(x^*(\tau), u^*(\tau), \tau) d\tau$
subject to
- $\dot{x}^*(\tau) = f(x^*(\tau), u^*(\tau), \tau)$
- $x^*(t) = x(t)$
- $x_{min} \leq x^*(\tau) \leq x_{max}$
- $u_{min} \leq u^*(\tau) \leq u_{max}$

ただし，$\tau (t \leq \tau \leq t+T)$ は変数，$x^*(\tau), u^*(\tau)$ は変関数である．目的関数は，終点コスト ϕ と区間コスト L によって構成され，ある時刻 t における状態 $x(t)$ が与えられたとき，時刻 $t+T$ と評価区間 $[t, t+T]$ において評価関数を最小化することを表している．制約式中の $x^*(\tau)$ 及び $u^*(\tau)$ は，状態値と操作量である．

現在の時刻を t_1 としたときの MPC の例を図 2 に

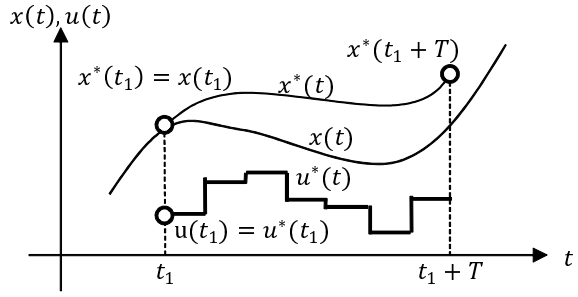

図 2 時刻 t_1 におけるモデル予測制御
Fig. 2 A model predictive conrtrol scheme at time t_1.

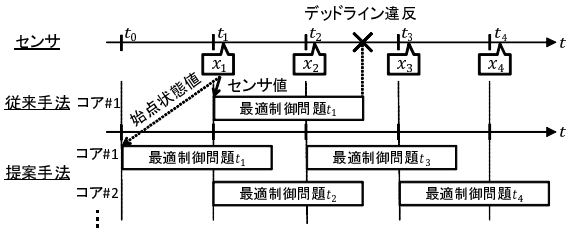

図 3 投機実行の概観図
Fig. 3 Overview of our speculative execution.

示す．時刻 t_1 の時の状態値 $x(t_1)$ を与え，最適制御問題を解くことで操作量 $u^*(t)$ を得る．実際の操作量は，初期値のみを用いて $u(t_1)=u^*(t_1)$ を与える．以降，MPC はサンプリング周期毎に最適制御問題を解き，解の初期値を操作量として与える．

3. 提 案 手 法

3.1 概　　要

提案手法は，MPC における最適制御問題をメニーコアで投機実行し，実時間 MPC を実現する．投機実行の概観を図 3 に示す．提案手法では，複数のプロセッサコアを活用して 1 サンプリング周期後の最適制御問題を予め実行することによる時間軸方向の並列性に着目する．最適制御問題 t_1 の実行には，時刻 t_1 に入力されるセンサ値 x_1 が必要である．そこで，x_1 の値を予測し，かつ，最適制御問題 t_1 の実行を時刻 t_0 から開始する．本稿では，予測した状態値を**始点状態値**と定義する．始点状態値の予測には，サンプリング周期が十分に小さい制御システムが対象の場合，制御対象の状態値が急激に変化しないという仮説の下，過去の最適制御問題の解を活用する．

提案方式では，始点状態値が制御対象の安定性に影響を与えるため始点状態値の予測精度が重要となる．そこで，複数コアを利用してセンサ予測値の近傍値を用いた投機実行も並列に行う．この場合，正しいセンサ情報が得られた時点でセンサ値と予測値の誤差が最も小さい最適制御問題の実行結果が選択される．

3.2 始点状態値の予測手法

最適制御問題を投機実行するために必要な始点状態値 $x(t_{n+1})$ の予測手法を提案する．$x(t_{n+1})$ の予測には，最も新しい最適制御問題の解を用いて $x(t_{n+1})$ を求める．さらに，$x(t_{n+1})$ は近傍を考慮した複数の値にすることで，予測精度を向上させる．始点状態値 $x(t_{n+1})$ とその近傍も含めた入力値候補の集合 $X_{t_{n+1}}$ を定式化すると，

$$X_{t_{n+1}} = \{x^*{}_{LRC}(t_n) \pm Rd | d, D \in \mathbb{N}^{\curlywedge}, 0 \le d \le D\} \quad (2)$$

と表される．ただし，$x^*{}_{LRC}(t_n)$，R，及び D は，最も新しい最適制御問題の解，丸めの単位，及び状態値の予測値と観測値のズレを示す**ばらつき**である．ばらつきは，システムの状態の観測値と予測値の差の絶対値の丸め値に対する指標である．例えば，$R=10^{-2}$ のとき，観測値と予測値の差の絶対値の丸め値が 0，0.01，0.02 であるとすると，ばらつきはそれぞれ，$D=0$，$D=1$，$D=2$ となる．すなわち，式 (2) において，$R*D$ は予測値の近傍範囲を表す．提案手法では，式 (2) で示す集合の要素数だけ計算機資源を使い，投機実行を行うので，計算資源と相関関係をもつ．したがって，計算資源と $X_{t_{n+1}}$ の予測精度のトレードオフ関係を定量化し，制御対象の特性に応じた適切な資源配分を行うことが重要である．

3.3 投機実行の性能モデル

提案手法の有効性を示すために，投機実行の性能をモデル化する．まず，予測値と観測値が一致することを**予測ヒット**と定義し，シミュレーション時間当たりの予測ヒットの割合を**予測ヒット率**と定義する．本手法で始点状態値の予測手法を適用した場合の投機実行の性能をモデル化すると，予測ヒット hit の条件は，

$$hit(t,R) = \begin{cases} 1 & (Round(x_{\mathrm{diff\_min}}(t),R)=0) \\ 0 & (otherwise) \end{cases} \quad (3)$$

$$x_{\mathrm{diff\_min}}(t) = \min_{x_{pred}(t)\in X_t} |x_{pred}(t) - x_{obser}(t)| \quad (4)$$

と表される．ただし，R，$x_{\mathrm{diff\_min}}(t)$ は，離散化による丸めの単位，時刻 t における観測値と予測値の差の絶対値が最も小さいものである．

式 (3) を用いて，予測ヒット率 hit_rate は，

$$hit\_rate(N,R) = \frac{\sum_{t=0}^{N} hit(t,R)}{N} \quad (5)$$

と表される．ただし，N は総サンプリング数である．総サンプリング数は，総シミュレーション時間をサンプリング周期で割った数である．

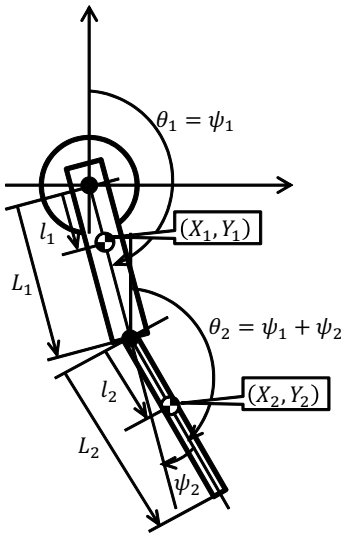

図 4 アーム型振り上げ振り子 (ψ_i:相対角，θ_i:絶対角)
Fig. 4 Arm typed pendulum swing-up control system (ψ_i: relative angle, θ_i: absolute angle).

4. 実　　験

4.1 実験環境

提案手法の有効性を示すために，MPC プログラムに対して始点状態値と観測値の比較，及び予測ヒット率を調査した．対象の MPC は，大塚ら[5] が作成したアーム型振り上げ振り子のシミュレーションプログラムを使用した．本稿では，状態方程式と最適制御問題の概要について述べる．定式化に関する詳細は[5] を参照されたい．シミュレーションのために用いた計算機の Linux kernel のバージョンは 2.6.32，CPU は Intel Xeon5670 2.93GHz である．本実験では，総ステップ数は $N = 9980$，丸めの単位は $R = 10^{-2}$，計算資源の単位はプロセッサコア数とした．

アーム型振り上げ振り子の模式図を図 4 に示す．2 つのアームが鉛直下方向 ($\theta_1 = \theta_2 = \pi$) で静止している状態を初期状態とし，両アームが鉛直上方向 ($\theta_1 = \theta_2 = 0$) で静止している状態を目標状態とする．まず，アーム型振り上げ振り子の状態方程式を定式化する．各リンクの重心位置は，

$$\begin{cases} X_1 = l_1 \sin\theta_1 \\ Y_1 = l_1 \cos\theta_1 \end{cases}, \begin{cases} X_2 = L_1 \sin\theta_1 + l_2 \sin\theta_2 \\ Y_2 = L_1 \cos\theta_1 + l_2 \cos\theta_2 \end{cases} \tag{6}$$

であり，振り子全体の運動エネルギー W，位置エネルギー U，及び損失エネルギー D はそれぞれ，

$$\begin{cases} W = \sum_{i=1}^{2} \left\{ \frac{1}{2} m_i \left(\dot{X_i}^2 + \dot{Y_i}^2 \right) + \frac{1}{2} J_i \dot{\theta_i}^2 \right\} \\ U = \sum_{i=1}^{2} m_i g Y_i, D = \sum_{i=1}^{2} \frac{1}{2} \mu_i \dot{\psi_i}^2 \end{cases} \tag{7}$$

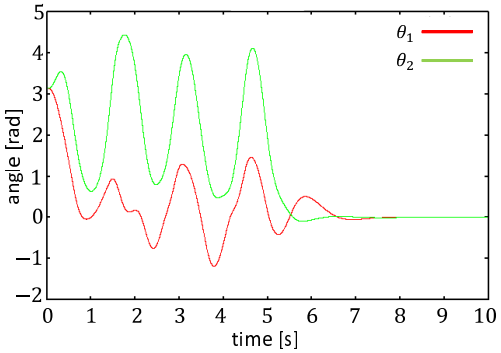

図 5 シミュレーション結果
Fig. 5 The angle of arms in experimental result.

と表される．ただし，m_i：各リンクの質量，J_i：各リンクの重心まわりの慣性モーメント，g：重力加速度，μ_i：各リンクの粘性摩擦係数である．運動方程式を基に MPC の最適制御問題に定式化すると評価関数 J は，

$$J = \frac{1}{2} x^T(t+T) S_f x(t+T) + \int_t^{t+T} \left(\frac{1}{2} x^T(\tau) Q_x(\tau) + \frac{r_1}{2} u^2(\tau) - r_2 v(\tau) \right) d\tau \tag{8}$$

と表される．ここで，$x = \left[\theta_1\ \theta_2\ \dot{\theta_1}\ \dot{\theta_2}\right]^T$，各リンクの絶対角を根元から順に θ_1, θ_2，S_f, Q は準正定行列，r_1, r_2 は正の実数である．評価区間の長さ T は $0.5[s]$，サンプリング間隔は $1[ms]$ とする．

次に制約条件は，

$$u^2 + v^2 - {u^2}_{max} = 0 \tag{9}$$

と表される．ここで，u は制御入力であるモータへの入力電圧であり，u の大きさは $|u| \leq {u^2}_{max}$ という拘束条件が課されるとする．最大入力電圧 u_{max} は 2.5[V] とする．さらに，v は制約式のために新たに導入した仮想的な入力である．

4.2 実験結果

まず，シミュレーションの結果を図 5 に示す．図 5 では，モータによって制御された各アームの絶対角を時系列で示している．今回の制御では，振り子を数回振ることで最終的に鉛直上向きで静止する振る舞いを示す結果となっている．

アーム部と振り子部の観測値と予測値の比較結果を図 6 に示す．予測値 $x(t_{n+1})$ には，プログラムの制約より現在時刻から 0.02[s] 後の状態の予測値を使用した．例えば，時刻 1.02[s] においては，時刻 1.00[s] に予測した状態値を予測値として扱い，時刻 1.02[s] の観測値の差の絶対値が図 6 にプロットされる．

次に，プロセッサコア数に対する予測ヒット率を図

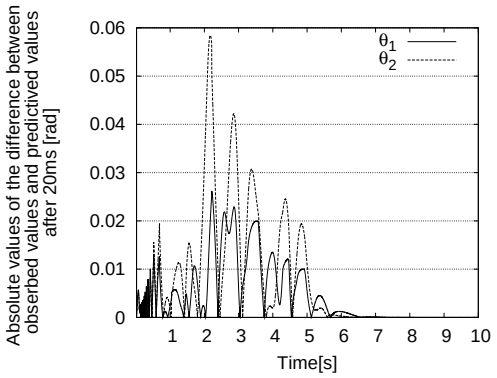

図 6 各時刻における 20ms 後の予測値と観測値との差の絶対値
Fig. 6 Absolute values of the difference between observed values and predicted values after 20ms for each sampling period.

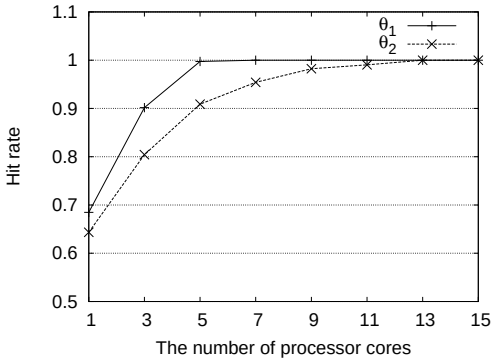

図 7 プロセッサコア数に対する予測ヒット率
Fig. 7 Hit rate of prediction for the number of processor cores.

7 に示す．プロセッサコア数の増加に伴い近傍の範囲が拡大することにより，予測ヒット率が向上する結果となった．図 7 より，θ_1 では 7 コア，θ_2 では 13 コアで予測ヒット率が 100%に達することを確認した．従って，本実験においては，プロセッサコア数が 13 の時，最適制御問題を投機実行することが可能である．

5. 関 連 研 究

本節では，MPC プログラムの高速化の関連研究をまとめる．MPC 高速化の研究は，ソフトウェア及びハードウェアからのアプローチが存在する．

ソフトウェアによるアプローチでは，まず，偏微分方程式の定常解が算出可能な定式化を行う手法がある．しかしながら，定常解を求める方法は，適用可能な制御システムが限定的であるという問題点がある[6]．定常解を求める他に，最適制御問題を解くための数値計算手法を高速化する研究が多数行われている[1),7)]．これらの研究では，数値計算手法を工夫することで，計算量を削減する手法が提案されている．

ハードウェアによるアプローチでは，並列処理によって高速化を図る研究が行われている[8)]．しかしながら，制御向けプログラムは逐次処理部分が多く存在するため，効果的な並列化が行われていない．

6. おわりに

本稿では，メニーコアプロセッサにおける実時間モデル予測制御のための投機実行法を提案した．提案手法では，最適制御問題を投機的に実行するために必要な始点状態値の予測手法の提案，及び投機実行の性能をモデル化した．始点状態値の予測手法を基に，MPC で定式化したアーム型振り上げ振り子制御に対して，始点状態値と観測値の比較，及び予測ヒット率を調査した．計算機実験により，計算資源であるプロセッサコア数が 13 個の時，予測ヒット率が 100%になることを確認した．更に，プロセッサコア数と予測ヒット率の間にトレードオフ関係が存在することを実験的に明らかにした．今後は，本稿で提案した予測ヒット率を基にリアルタイム制約を検証する必要がある．さらに，メニーコアプロセッサへの実装，評価も行う．

参 考 文 献

1) Wang, Y. and Boyd, S.: Fast Model Predictive Control Using Online Optimization, *IEEE Trans. Control Systems Technology*, Vol. 18, No. 2, pp. 267 –278 (2010).
2) Xu, H., Tanabe, J., Usui, H., Hosoda, S., *et al.* : A low power many-core SoC with two 32-core clusters connected by tree based NoC for multimedia applications, *Symposium on VLSIC*, pp. 150 –151 (2012).
3) Bell, S., Edwards, B., Amann, J., Conlin, R., *et al.*: TILE64 - Processor: A 64-Core SoC with Mesh Interconnect, *Proc. ISSCC*, pp. 88 –598 (2008).
4) Ferreau, H., Lorini, G. and Diehl, M.: Fast nonlinear model predictive control of gasoline engines, *Proc. IEEE International Conference on Control Applications*, pp. 2754 –2759 (2006).
5) 大塚敏之: モデル予測制御 (発展編, 特集：初学者のための図解でわかる制御工学 II), システム/制御/情報 : システム制御情報学会誌, Vol. 56, No. 6, pp. 310–312 (2012).
6) 大塚敏之: 非線形最適制御入門, コロナ社 (2011).
7) Ohtsuka, T.: Continuation/GMRES method for fast algorithm of nonlinear receding horizon control, *Proc. IEEE Conference on Decision and Control*, Vol. 1, IEEE, pp. 766 –771 (2000).
8) Gu, R., Bhattacharyya, S. and Levine, W.: Methods for efficient implementation of Model Predictive Control on multiprocessor systems, *IEEE International CCA*, IEEE, pp. 1357 –1362 (2010).