MAPLE core におけるレシーブレジスタ割り当て手法

岩井 啓輔 †　森村 知弘 ‡　阿部 剛 ‡　天野 英晴 ‡

† 防衛大学校
‡ 慶應義塾大学

MAPLE core の備えるレシーブレジスタ通信機構によるプロセッサ間データ転送は，軽量バリア同期を組み合わせることにより，低コストかつ実装の容易なハードウェアで効率の良いプロセッサ間通信を実現可能で，オンチップマルチプロセッサを想定して実装された．

この方法は，レジスタの数に制限があるため，コンパイラによりいかに効率良く割り付けを行うかが性能向上に大きく影響を及ぼす．本論文では，レシーブレジスタを効率よく使用するための割り当てアルゴリズムを提案する．

大規模数値計算プログラムの近細粒度並列処理に当手法を適用した結果，人手による最適化には及ばないが 4PE まで性能向上が達成された．

Receive Register allocation algorithms on MAPLE core

Keisuke IWAI†.　Tomohiro MORIMURA‡.　Tsuyoshi ABE‡.　Hideharu Amano‡

†National Defense Academy
‡Keio University

A processing element MAPLE core has a light-weight processor-to-processor communication mechanism using special registers called Receive Register and a light weight barrier synchronization for effective near fine grain parallel processing. Although a high speed data transfer can be performed with Receive Registers, an overwriting problem occur because of the register number limitation.

In this report, we propose Receive Register allocation algorithms which can use Receive Registers without overwriting the communication data. Clock level simulation results show that using the proposed allocation algorithm, the performance is improved with the near fine grain parallel execution.

1. まえがき

マルチプロセッサシステムにおいて，様々な粒度の並列性を有効に利用する自動並列化手法とアーキテクチャサポートは重要である．当研究室では，マルチグレイン自動並列処理向けマルチプロセッサアーキテクチャである ASCA システムを提案した．この ASCA の要素プロセッサとなる専用プロセッサコア MAPLE core[1] は，近細粒度並列処理を効率よく実行することが可能な構成になっている．その特徴の一つは，あるプロセッサの汎用レジスタ（GPR）から，直接他プロセッサの受信用レジスタ（レシーブレジスタ）にデータ転送を行なうことが可能なプロセッサ間通信機構を備えていることである．この通信機構（レシーブレジスタ通信機構）は，簡単なハードウェアと局所的なハンドシェークで実装可能であり，並列処理時のデータ転送オーバヘッドを効果的に低減することができるが，送信データの上書きを生じる可能性がある．本報告では，コンパイラの静的スケジューリングにより，軽量バリアを組み合わせて使用し，上書き問題を解決するレシーブレジスタ割り当て法を提案する．

2. MAPLE core とレシーブレジスタ通信機構

MAPLE core はマルチグレイン並列処理向けマルチプロセッサシステムである，ASCA の要素プロセッ

サのプロセッサコアである．MAPLE core は，Rohm 社の 0.35μm の CMOS 上に実装され，実配線シミュレーションの結果，80MHz で動作することがわかっている．必要ゲート数は全体で約 20 万であり，今回問題とするレシーブレジスタ通信機構と軽量バリア同期は合わせても 6000 ゲート程度で実装されている [3]．

MAPLE core は，近細粒度を効率よく行うために以下のような特徴を持つ．

- コンパクトなハードウェア
- 高速な浮動小数点演算
- 軽量通信機構
- 同期／無同期モード

特に本手法で使用する軽量通信機構である，レシーブレジスタについて紹介する．

2.1 レシーブレジスタ通信機構の概念

レシーブレジスタ通信機構によるプロセッサ間データ転送の概念図を図 1に示す．

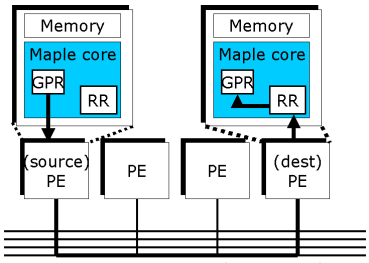

図 1 レシーブレジスタ通信機構

MAPLE core は，GPR の他に受信専用のレジスタ (レシーブレジスタ) を持つ．近細粒度並列処理時など，主に小さいサイズのデータ転送が頻繁に起こる場合は，ネットワークを介して，自 MAPLE core の GPR の内容を直接相手 MAPLE core のレシーブレジスタに転送する．受信側は受信命令を発行して，受信したデータを汎用レジスタに移動することで受信動作が完了する．データの到着は，レシーブレジスタに到着を示す valid bit を持たせることにより実現する．レシーブレジスタにデータが到着していなければ受信側プロセッサはストールする．MAPLE core は，複数の MAPLE core を同一チップ上に実装することを想定しており，エレメント間の接続は強力なクロスバを用いる．

この方式は，受信のみ専用レジスタを持たせて，同期と受信を一体化しており，受信処理は時プロセッサ内のレシーブレジスタの状態を見る局所的な信号で実現可能である．

一方で，送信は受信レジスタの valid bit の状態に関わらず実行できる．これは，同一チップ上を想定するとはいえ，受信側 PE のレシーブレジスタの状態をチェックして送信側 PE の動作を停止させることが，実装上困難であることによる．したがって，この方式は，送信側がデータを上書きしないように制御する必要があり，これが困難であることが，今までこの方法が採用されなかった大きな原因になっていると考えられる．

今回はこの問題に対して，コンパイラの管理により，軽量バリアと組み合わせることにより制御を行う方法を提案する．以下にレシーブレジスタの操作プリミティブを紹介しておく．

- `sendr/sendri` ノンブロッキング送信命令．それぞれレジスタオペランド，即値オペランドをとる
- `rcvr/rcvri` ブロッキング受信命令．それぞれレジスタオペランド，即値オペランドをとる

3. レシーブレジスタ割り当て手法

3.1 レシーブレジスタ割り当ての基本方針

3.1.1 Barrier 命令の導入

MAPLE core のレシーブレジスタ通信における送信命令（以下 Send）は，転送先レジスタの valid bit の状態を確認できないため，消費していないデータを書き潰してしまう恐れがある．そこで，再割り当て時には何らかの方法でレシーブレジスタ内のデータの消費を確認しなくてはならない．ここでは，MAPLE core の軽量 Barrier 同期を用いてレシーブレジスタのデータの消費を保証する方法を提案する．

この方法では，図 2 に示すように，すべてのレシーブレジスタに一度データが書き込まれ，それが受信命令（以下 Recv）によって消費されると，全 PE（MAPLE core) で Barrier を発行する．この図では，レシーブレジスタ数を 4 としている．書き込み側は，上書きを起こさないように，決まった量のデータを書き込んだ時点で Barrier を発行する．これにより，Barrier が成立するまで Send は発行されなくなるので，レシーブレジスタの上書きはおこらない．

```
     PE0          PE1          PE2
                 recv r0←send pe1 r0
send pe1 r1→recv r1
send pe1 r2→recv r2
                 recv r3←send pe1 r3
Barrier_____Barrier__Barrier____
    再使用       recv r0←send pe1 r0
```

図 2 Barrier 同期と組み合わせたレシーブレジスタ使用法 (レシーブレジスタ数=4)

3.1.2 RR サイクルの概念の導入

上記に示した方法では，すべてのレシーブレジスタのデータがいったん消費されると Barrier が挿入され，次のデータを受け入れることを繰り返す．ここで RR サイクルという概念を導入する．

> あるプロセッサにおいて，すべてのレシーブレジスタが未使用の状態からすべてのレシーブレジスタについて転送が行なわれ，さらにすべての Recv が発行されるまで（すなわち転送されたデータがすべて消費されるまで）を **1RR サイクル**と呼ぶ．(図 3)

すなわち，あるプロセッサでレシーブレジスタと同じ数だけ（違うレシーブレジスタに対する）の Recv が発行されることが，1RR サイクルに相当する．ここで RR サイクルは各 PE それぞれにおいて独立に，それぞれの発行する Recv により定義されることに注意されたい．

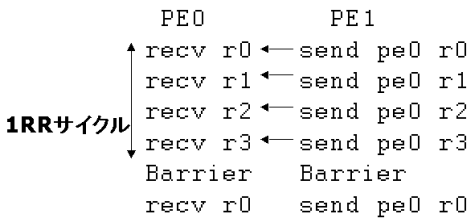

図 3 RR サイクル

3.1.3 Recv と Send の発行順

このアルゴリズムでは無駄なストールを避けるため Recv はデータが必要になる直前に挿入する．このようにすると，Recv はデータの使用順に並び，レシーブレジスタはサイクリックに用いることができる．

まず，Recv は使用される順番に並び替えられたのでそれに番号を振る．これを recvNo と表記する．ある Recv についての RR サイクル番号は，

$$recvNo/\text{レシーブレジスタ数} \quad (\text{小数点以下切り捨て})$$

で与えられる．次に，Send/Recv は，この RR サイクルと Recv の順番により以下のように識別する．

> Recv(src, recvNo, CNo)
> src: ソース PE 番号，recvNo: この Recv がその PE 内で発行される番号，CNo: RR サイクル番号
> Send(dest, recvNo, CNo)
> dest: 出力先 PE 番号，recvNo: この Send に対応する Recv の recvNo，CNo: 対応する Recv の RR サイクル番号

ここで，Send を識別する情報はすべて対応する Recv のものであることに注意されたい．

さて，Send は，計算と通信のオーバラップを考慮すると，転送すべきデータが用意ができたらすぐに発行することが望ましい．しかし，レシーブレジスタを使用する場合は，データが用意できた時点で Send を発行すると，問題サイズによってはレシーブレジスタ数より多くのデータの Send を発行してしまう．このことを避けるため，レシーブレジスタの利用可能数と，Send するデータの使用される順（すなわち recvNo と CNo）を用いて Send の発行タイミングを調節する．

3.1.4 Barrier の挿入とオーナ

一つの PE のコードには通常 Send も Recv も含まれる．それらはそれぞれ独立した RR サイクルを持つが，一つの PE の命令列にこれら RR サイクルが混在する．そこで，これらを区別するために，RR サイクルにどの PE の Recv によって定義されたものかを識別する番号を付け加える．この番号は RR サイクルを定義した PE の番号で，これを RR サイクルのオーナと呼ぶ．これにより RR サイクルは次のようにカラーリングされる．

> RR サイクル (Owner, CNo)
> Owner: RR サイクルのオーナ，CNo: RR サイクルの番号

Barrier はオーナ PE にかかわらず，RR サイクルの変化時に挿入される．Barrier についても同様の記述を用いてカラーリングする．

> B(Owner, CNo)
> Owner: RR サイクルのオーナ，CNo: RR サイクル番号

図 3 で示した RR サイクルについて新しく定義した Send/Recv/Barrier の表記法を用いると図 4のようになる．

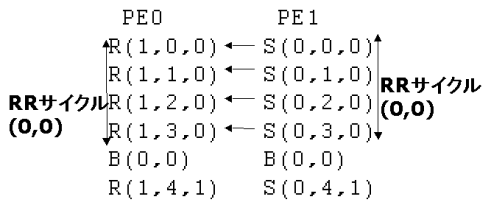

図 4 RR サイクル（別表記）

以下，スタティックスケジューリング中に，上記の原則と記法に基づいてレシーブレジスタ割り付けを行なう方法を具体的に示す．

3.2 レシーブレジスタ割り当て

今回提案するレシーブレジスタの割り当ては近細粒度タスクに対するスタティックスケジューリングであればほとんどのものとそのまま組み合わせることが

可能であるが，ここでは説明のため，CP/DT/MISF 法[4] と組み合わせる．

3.2.1 CP/DT/MISF 法

CP/DT/MISF 法は実行時間最小マルチプロセッサスケジューリング問題のヒューリスティックな解法として提案されたもので，近細粒度タスクの並列処理に用いられる．そのアルゴリズムを以下に示す．

(1) 各タスクから終了までの最長パス長（クリティカルパス/CP）を計算する．
(2) あるスケジューリング時点において実行可能タスクのうち CP が最大のタスクを，アイドルプロセッサへ割り当てた時のデータ転送時間を計算する（最初の割り当て時は 0 とする）
(3) 手順（2）で計算されたデータ転送のうち，最小のデータ転送時間を必要とするタスクのプロセッサへの割り当てを 1 つ選ぶ．その際，同一のデータ転送時間を与える割り当てが 2 つ以上あれば，直接後続タスク数のもっとも多いものを選ぶ．レディタスク／アイドルプロセッサがあれば（2）から繰り返し，なければ（4）へ進む．
(4) 少なくとも 1 つのプロセッサが実行を終了するまでスケジュール時間を進め，レディタスクがあれば（2）へ，なければ（4）を繰り返す．すべてのプロセッサの実行が済めば終了．

レシーブレジスタ割り当てで重要なのは実際にタスクがプロセッサに割り当てられたタイミングである．すなわち手順 3 において，タスクがプロセッサに割り当てられ，タスク間の依存により，転送すべきデータとプロセッサが決定する．ここで使用するレシーブレジスタを決定する．

3.2.2 RR の割り当て

CP/DT/MISF 法の手順の 3，割り当て時において以下の手順を加える．

(1) 必要なデータ転送 (Recv) が決定（タスクの割り当てが確定すれば Recv とそれに対応する Send は自動的に決定する）
(2) 受け側はレシーブレジスタ確保を図り，レシーブレジスタが全て用いられているならば，(すなわち自 PE がオーナの RR サイクルが変化する）手順（4）へ．確保できたならば，そのレシーブレジスタへの必要なデータの Recv 命令を割り当てられるタスクの前に挿入
(3) 転送元 PE の現在実行中のタスクの後に Send 挿入，次のスケジュールへ
(4) レシーブレジスタがいっぱいであったならば，全プロセッサの現スケジューリング時間にに Barrier 命令を挿入（実行中のタスクがあるなら終了後に挿入）/Recv を発行した PE がオーナの RR サイクルの増加/受け (オーナ) プロセッサのレシーブレジスタの全エントリを解放し，改めて手順（2）へ．

この作業をスケジューラに追加することによりレシーブレジスタ転送を使用することができる．これを CP/DT/MISF(RR-Barrier) 法とここでは呼ぶ．ここで挿入された Send のほとんどは，後述の方法によって最適化されより早い発行タイミングに変更される．

3.2.3 割り当ての特徴

この方法には以下の特徴がある．

a)Send のタイミング

通常のメッセージパッシング処理では，送信データが用意されると，データ転送のオーバヘッドを隠蔽するため，なるべく早く転送を行なうが，ここではレシーブレジスタ数の制限により，送信タイミングを通常より遅らせることとなる．

そこで，上書きの起こるような過剰な Send の先行発行を制限したうえで，できうる限りの先行発行を行なう最適化を組み込む．これは Barrier と Send の Owner や RR サイクルといったカラーリング情報を用いることで容易に実現でき，ほとんどの Send が早いタイミングで発行される．b)Barrier による一斉同期 Barrier により，全プロセッサで同時に，あるプロセッサのレシーブレジスタが使用可能になったことを知ることができるが，MAPLE core ではプリミティブな Barrier しか実装されていないため，実際に有効な Barrier の発行タイミングには幅があるのに，実際には Barrier を挿入する一箇所を決定しなくてはならない．Barrier 挿入タイミングによっては無駄なストールが発生する可能性があるため，本手法では，スタティックスケジュール時の実行見積もりで，全プロセッサが Barrier 待ち時間が少ないようなタイミングで Barrier を挿入することで，損失の低減を図っている．

3.3 Barrier を使用しない同期

PE 数が少ない場合，Barrier を使用せずに，レシーブレジスタの一部を同期用に用いて前述と同様の処理を実現することができる．

この方法では，Recv がブロッキング受信であることを利用して，レシーブレジスタの消費を知らせる．RR サイクルの変化時にオーナプロセッサは，Barrier に代えて Send を他のすべてのプロセッサに発行し，そのほかのプロセッサは，対応する RR サイクル変化

時に，対応する Recv を発行する．このことで，RR サイクルの同期をとることができる．

図 5では通常の Send/Recv と，RR サイクルの同期用の Send/Recv を区別するために，同期用 Send を SS(destPE, CNo)/SR(OwnerPE, CNo) と表記する．それぞれ，destPE は出力先 PE 番号，OwnerPE はその RR サイクルのオーナ，CNo は RR サイクルの番号である

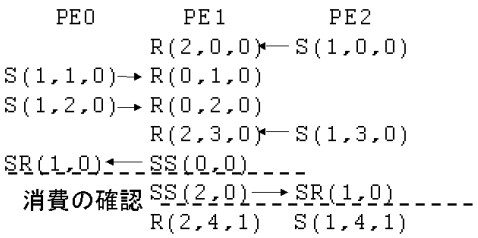

図 5　レシーブレジスタを利用した RR サイクル同期

4. 性能評価

4.1 評価環境

MAPLE core チップは既に実装されており，そのモジュールは全て Verilog-HDL によって記述されている．そこで，今回の評価は，設計データに基づく RTL シミュレーションで実プログラムを元にしたベンチマークを使用して行なっている．動作周波数は 50MHz とした．

4.1.1 MAPLE cluster

評価には，オンチップマルチプロセッサを想定したマルチプロセッサ MAPLE cluster を用いる．この構成では，各 MAPLE core をクロスバで接続し，レシーブレジスタ通信機構を用いたプロセッサ間通信を行なう．クロスバは，1 クロックで転送可能で，調停も 1 クロックで行なわれる．ピン数の制限からレシーブレジスタの転送ビット幅は 32 ビットとしている．またレシーブレジスタ数は各 MAPLE core について 16 とした．

4.1.2 評価用コード作成

評価用のコードの生成は以下の手順で行った．

近細粒度レベルでの評価であるため，プログラムから近細粒度並列処理を行なうブロックを抜き出し，マルチプロセッサ構成の MAPLE core 用の近細粒度自動並列化コンパイラプリプロセッサを用いて並列化 C を生成した（今回提案したレシーブレジスタ割り当てアルゴリズムはこのプリプロセッサ内に組み込む.）．その後 MAPLE core 用 C コンパイラバックエンドにより命令イメージファイルを作成した．全命令とデータは各 MAPLE core のキャッシュに搭載される．C コンパイラバックエンドのオプションは -O3 を用いた．

今回近細粒度並列化コンパイラが使用したスケジューリングアルゴリズムは CP/DT/MISF 法ではなく，よりデータ転送コストを重視した DT/CP 法 [5] を採用した，DT/CP(RR-Barrier)，および，レシーブレジスタのみで同期を取る DT/CP(RR-RR) を用いる．

4.1.3 評価プログラム

4.2 数値計算プログラム (fpppp) による性能評価

fpppp は SPEC FP 95 ベンチマークに収録されている，量子化学計算を行なうアプリケーションである．fpppp は幾つかのサブルーチンから構成されるが，実行時間の大部分を担当するサブルーチン `fpppp` は大規模な近細粒度タスクをもち，レシーブレジスタの再割当のアルゴリズムの検討に向いている．近細粒度タスク数は 333 個であり，そのタスクはほとんどが double 型の演算である．ただし除算は存在しない．データセットは小さく，全データをキャッシュに収めることが可能である．

4.2.1 Barrier と組み合わせた自動レジスタ割り当て

（スケジューリングも含め）人手を用いて並列化した場合と，Barrier を利用したレシーブレジスタの再割当，DT/CP(RR-Barrier) 法を用いて並列処理をした場合の並列効果を示す．図 6に MAPLE core を 1 ～ 5 台にしたときの実行時間を示す．

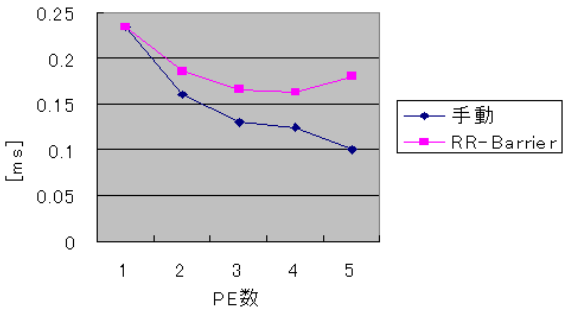

図 6　fpppp の計算時間

自動並列化を用いると 4 台で頭打ちになり，5 台では実行時間が増加している．これは人手による並列化に比べて Barrier のオーバヘッドがプロセッサ数の増加と共に影響してくるためと思われる．

人手による並列化でも Barrier を用いているが，5 台まで性能を伸ばしている．これはタスクグラフをその形状から直観的に切り分け，Send/Recv のタイミングを設定して，効果的に Barrier が取れるようにスケジューリングしており，無駄な Barrier によるプロセッサのストール時間を削ることに成功している．

これらの結果から，今回のアルゴリズムではレシーブレジスタと Barrier の組み合わせによるオーバヘッドが人手で行う場合に比較すれば大きくなっていることを意味している．

4.2.2 レシーブレジスタのみを利用した並列処理

次に，Barrier を用いず，レシーブレジスタのみを用いた DT/CP(RR-RR) 法で並列化した結果を示す．16 のレシーブレジスタのうち，PE 数-1 のレシーブレジスタが同期のために予約される．また，fpppp ではすべて倍精度データの転送が行なわれるため，1 つのデータに付き 2 つのレシーブレジスタを使用する．1RR サイクルで転送できるデータ数を表 1に示す．

表 1　1RR サイクルで転送できるデータ数

プロセッサ数	予約レジスタ数	転送可能データ数
2	1	7
3	2	7
4	3	6
5	4	6

実行結果を図 7に示す．

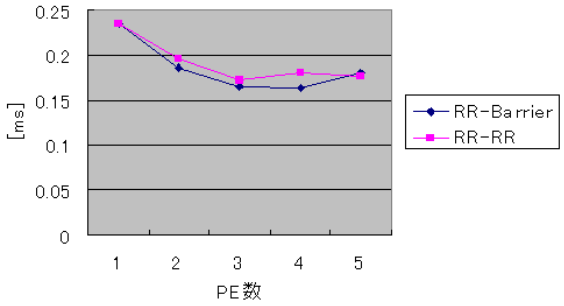

図 7　fpppp の計算時間 (RR-Barrier/RR-RR)

基本的に DT/CP(RR-RR) 法のほうが微妙に性能が悪くなっている．この手法ではプロセッサ数が増えると，RR サイクルの同期用にレシーブレジスタが使用され，使用可能レシーブレジスタが少なくなるため，プロセッサ数が少ないほうが有利のはずである．DT/CP(RR-Barrier) 法ではプロセッサ数が多くなると，無駄な Barrier を発行する機会が増えてしまい，性能向上が頭打ちとなる．一方で，DT/CP(RR-RR) 法でもプロセッサ数が増えると使用可能レシーブレジスタ数が減り，1 RR サイクルで転送できるデータ量が減るので傾向が同じとなる．

しかし，全体を通して DT/CP(RR-Barrier) 法の方が良い傾向を示すことから，このアプリケーションでは，最大数のレシーブレジスタを利用した通信と軽量 Barrier を組み合わせた方が性能が良いといえる．

5. むすび

MAPLE cluster におけるレシーブレジスタ通信機構のコンパイラによる制御方法を提案した．レシーブレジスタ通信機構は厳密に送受信の管理を静的に行なわなくてはならないが，今回のアルゴリズムではコンパイラにより簡単にレシーブレジスタ通信機構を使用できることを示した．

近細粒度並列処理を行なうアプリケーションの実行性能を評価した結果，人手による並列化には劣るものの，4PE 程度のクラスタではコンパイラの自動制御で並列効果を得ることができることが確認された．

今後は，MAPLE core の実行時間の静的予測可能な構成を生かし，より正確なスケジューリングによりオーバヘッドを減らす手法を検討する．

謝　辞

本研究は、半導体理工学研究センター (STARC) の援助を受けている．

参考文献

1) T.Fujiwara, T.Kawaguchi, K.Sakamoto, K.Iwai, and H.Amano. Custom Processor for the Multiprocessor ASCA. In *Proc. of 6th I-ASTED Symposium on Applied Informatics*, Feb. 1998.
2) 鈴木 貴幸, 川口 貴裕, 岩井 啓輔, 天野 英晴. マルチプロセッサシステム ASCA 用 FPU チップ. 第 3 回システム LSI 琵琶湖ワークッショップ, pp. 223–226, November 1999. (ポスターセッション).
3) T.Abe, T.Morimura, T.Suzuki, K.Tanaka, M.Koibuchi, K.Iwai, and H.Amano. ASCA chip set: Key components of multiprocessor architecture for multi-grain parallel processing. In *Proc. of COOL Chips IV*, pp. 223–247, April 2001.
4) 笠原 博徳. 並列処理技術. コロナ社, 1991.
5) 笠原 博徳. マルチプロセッサシステム上での近細粒度並列処理. 情報処理, Vol. 37, No. 7, pp. 651–661, Jul. 1996.
6) SPEC Corporation. SPEC Web page (http://www.spec.org).
7) MEDIABENCH Home (http://www.cs.ucla.edu/ leec/mediabench/).