論理関数の高速な評価手法について

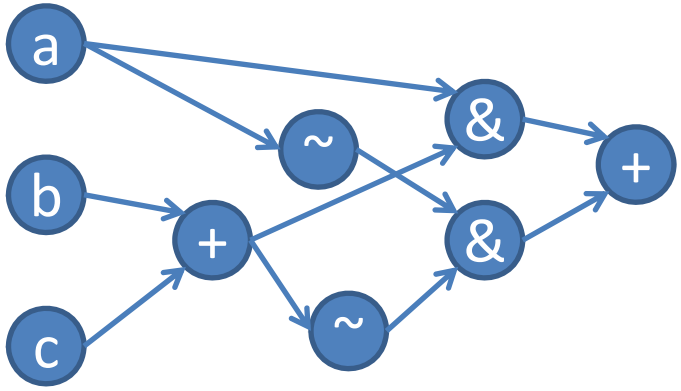
九州大学
松永　裕介

論理関数の評価

- 論理回路のシミュレーションでは古典的な話
  - 回路の要素数に比例した手間で実行可能
- 論理式の評価
  - 論理式の長さ（演算子数）に比例した手間で実行可能
- 真理値表
  - 変数の数に比例した手間で実行可能
- BDD
  - BDDの平均パス長（<変数の数）に比例した手間で実行可能

論理式と論理回路

- ほぼ同じ。違いは tree か DAG かということ。
  $a(b+c)+\bar{a}\bar{b}\bar{c}$

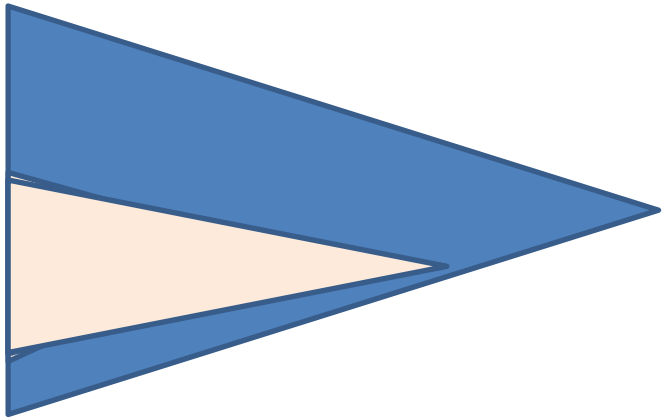


- 論理式も中間変数を導入すれば論理回路と同等の表現が可能(ブーリアンネットワーク)

論理回路の評価

- DAGを入力側からトポロジカル順にノード（演算子）を取出し、値を計算
- 通常はすべてのノードを評価する。
- 連続してよく似た入力の評価を行う場合には、差分だけを伝搬すればよい（イベントドリブン法）

差分の
ある入力

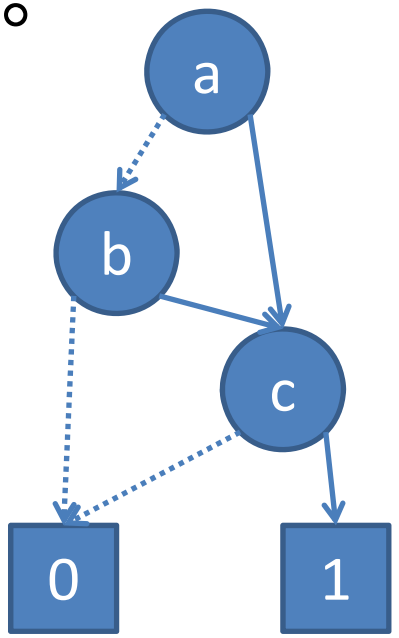


真理値表の評価

- アドレス計算に入力数に比例した手間が必要
- アドレスが確定すれば定数時間で答えがわかる。
- 常に、入力数の指数乗に比例したメモリ量を必要とする。
- 入力数が20程度なら可能性はある。

BDDの評価

- BDDの根のノードから入力変数の値に応じて0枝か1枝を選んでゆく。
- 定数節点にたどり着いたら終わる。
- 計算複雑度は根から葉までの
  平均パス長。
  全ノード数には依存しない。
- 理論的には最強だが。。。。



実験

- MCNC論理合成ベンチマーク回路、ISACS89テスト生成ベンチマーク回路
  - 数十～数百入力、～数千ノード
- ランダムに生成した入力を大量に評価、計算時間を計測

論理シミュレーション VS BDD

回路名	論理sim	BDD	BDD/sim.
C432	1.29	44.70	34.65
C499	2.05	439.50	214.39
C880	2.00	74.20	37.10
C1908	1.90	133.70	70.37
C3540	4.45	122.30	27.48
C5315	8.35	250.70	30.02
C7552	10.70	265.00	24.77
des	18.57	580.40	31.25
too_large	20.66	6.50	0.31
vda	3.11	84.80	27.27
s1196	2.23	67.90	30.45
s1423	2.99	147.70	49.40
s1494	2.64	52.10	19.73
s344	0.73	31.90	43.70
s35932	81.20	2311.90	28.47

大まかに言って64パタンの評価にかかる時間（μ秒）

論理シミュレーション VS BDD

回路名	論理sim	BDD	BDD/sim.
C432	1.29	44.70	34.65
C499	2.05	439.50	214.39
C880	2.00	74.20	37.10
C1908	1.90	133.70	70.37
C3540	4.45	122.30	27.48
C5315	8.35	250.70	30.02
C7552	10.70	265.00	24.77
des	18.57	580.40	31
too_large	20.66	6.50	

大まかに言って64パタンの評価にかかる時間（μ秒）

単純な比較では論理回路のシミュレーションに惨敗

なぜ？

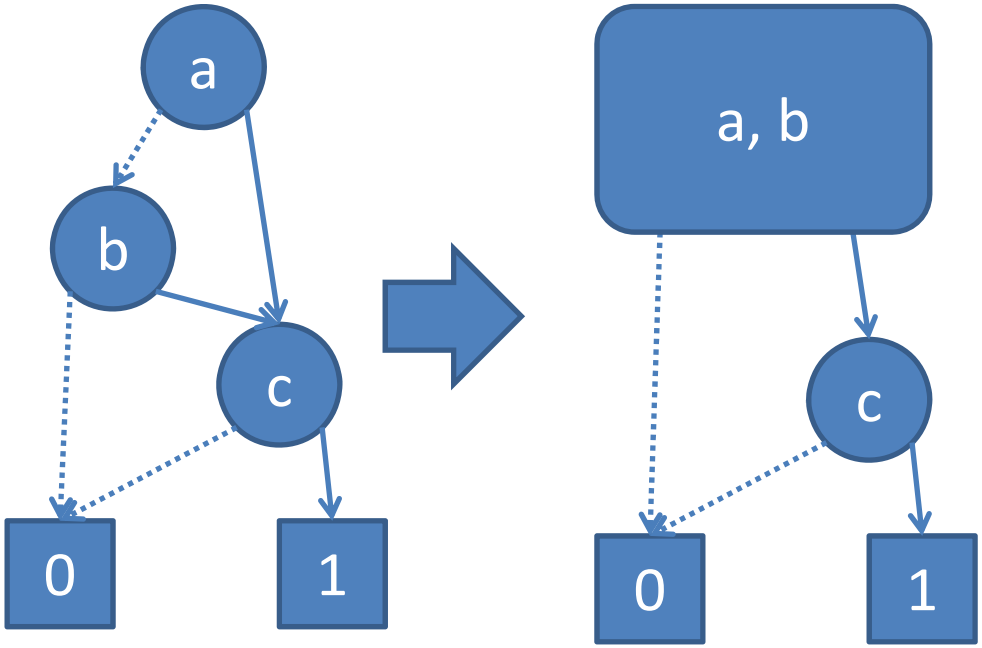
- ワードレベルの並列性
  - 論理演算は同時に異なる６４個のパタンを評価可能
  - BDDでは同時に1つの値に従った分岐しか評価できない。
- 複数の出力（論理関数）の評価に対する情報の共有
  - 論理回路の内部ノードの値は複数の出力の評価に用いられる。
  - BDDは各出力ごとに独立に評価を行う。
- メモリアクセスの違い
  - 論理回路はトポロジカル順に一度だけメモリアクセスが発生する。
  - BDDの広範囲に分布したランダムアクセスがキャッシュミスを頻発する。

BDDベースの手法の工夫

- 複数ノードのマージ
  - メモリアクセス回数の低減
- BDDの論理回路化
  - BDDのノードをマルチプレクサ（セレクタ）に置き換えることで、論理回路を構成可能
  - その回路を論理シミュレーションする。

BDDのノードマージ

- Kレベルのノードを1つのノードにマージ
- 2^k個の枝を持つ。



- 平均のメモリアクセス回数はｋ分の1なると期待できる。
- 実際にはそうでもない。

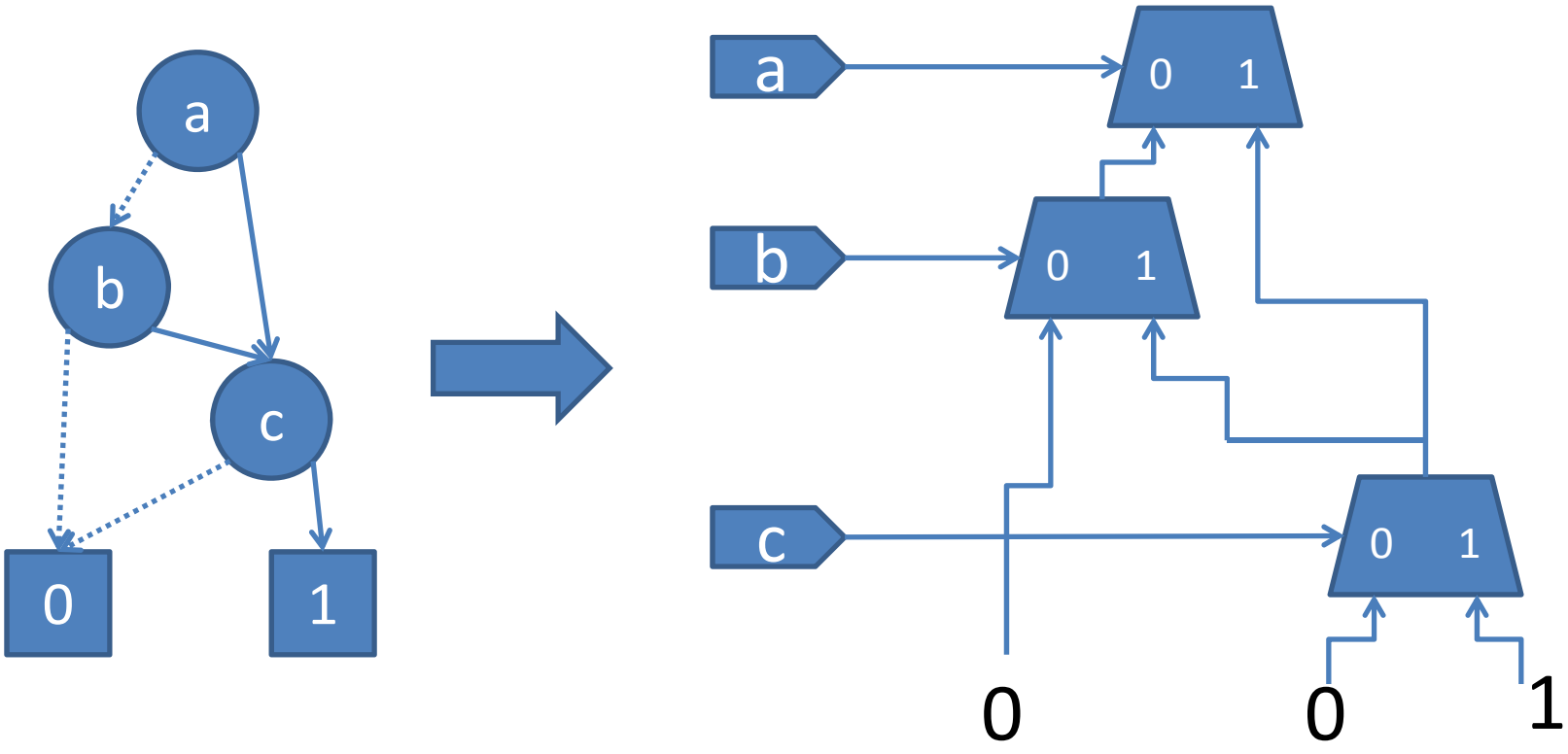
実験結果

回路名	BDD2 ratio	BDD10 rati
C432	0.99	0.88
C499	0.83	1.23
C880	0.85	0.87
C1908	0.84	0.79
C3540	0.90	1.02
C5315	0.93	1.05
C7552	0.87	0.83
des	0.97	1.13
too_large	1.15	0.78
vda	0.90	0.81
s1196	0.90	0.71
s1423	1.05	1.04
s1494	0.91	0.70
s344	1.10	1.09
s35932	0.80	1.49
s38417	1.16	1.97

- BDD2:
  隣接2レベルをマージ
- BDD10:
  隣接する10レベルをマージ

BDDの論理回路化

- ノード数とマルチプレクサ数は同じ。
- ワードレベルの並列性を活かすことができる。
- ただし、ふつうの回路より数倍～数十倍大きな回路ができる可能性がある。



実験結果

回路名	SIM	MPX2	MPX10	MPX2/SIM	MPX10/SIM
C432	1.29	5.00	8.40	3.88	6.51
C499	2.50	110.90	108.90	44.36	43.56
C880	2.00	17.80	36.00	8.90	18.00
C1908	1.90	23.30	25.20	12.26	13.26
C3540	4.45	139.40	236.80	31.33	53.21
C5315	8.35	9.00	13.70	1.08	1.64
C7552	10.70	11.50	18.20	1.07	1.70
des	18.57	15.10	21.50	0.81	1.16
too_large	20.66	1.80	2.70	0.09	0.13
vda	3.11	2.20	2.30	0.71	0.74
s1196	2.23	2.80	3.40	1.26	1.52
s1423	2.99	8.50	12.80	2.84	4.28
s1494	2.64	1.70	2.00	0.64	0.76
s344	0.73	0.70	0.80	0.96	1.10
s35932	81.21	43.70	50.90	0.54	0.63
s38417	65.30	1566.50	3258.60	23.99	49.90
s5378	8.31	10.60	14.50	1.28	1.74

まとめ

- 論理関数に対する小さな表現があればそれを用いるのがよい⇔BDDは論理関数の最小表現とは限らない。
- 分岐プログラム、表引き方式はワードレベルの並列性を活かせない。簡潔な論理式が最も望ましい。
- BDDが与えられたときに、簡潔な論理式を求める汎用的な手法があれば理想的⇒EDAにおける「論理合成」の理想形