# リアルタイムビートトラッキングシステムの並列計算機への実装

── ＡＰ１０００によるリアルタイム音楽情報処理 ──

後藤　真孝　　村岡　洋一

早稲田大学　理工学部

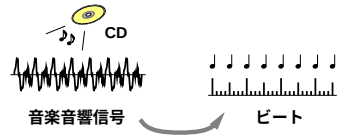




## １．はじめに

### ロ ビートトラッキングとは？

人間が音楽に合わせて手拍子を打つように
曲のビート（４分音符）の位置を認識する技術




### ロ なぜ重要なのか？

- 計算機による音楽認知モデルに不可欠
  - 音響信号に対する音楽聴取過程を計算機上で実現する上で重要
  - 西洋音楽を認識する上で基本的な能力の一つ
  - 計算機に他者の演奏を理解する能力を持たせる第一段階
- アプリケーション
  - 音楽に同期したリアルタイムＣＧの生成
  - ビデオ編集システム
  - オーディオ編集システム
  - ライブの照明制御



### ロ 音楽音響信号に対するビートトラッキングシステム（ＢＴＳ）

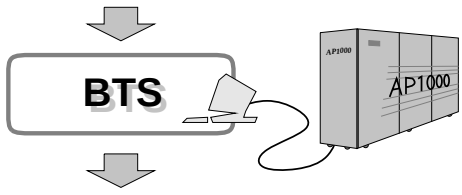
４分音符に相当するビートをリアルタイムに認識・予測

**音楽の音響信号**　（ロック・ポップスのＣＤ）

ドラムスを含む複数の楽器による演奏音

４／４拍子　テンポ７０～１８０（４分音符／分）

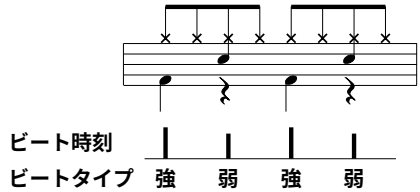
ロック・ポップスではテンポはほぼ一定




**ビート情報**　（４分音符の位置に相当）

ビート時刻・ビートタイプ・現在のテンポ

入力した音楽に合わせてリアルタイムに出力



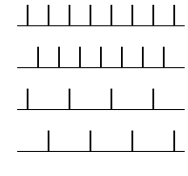



## ２．並列処理の必要性

計算量の多い処理をリアルタイムに実行するため

### ロ ビートトラッキングの難しさ

- ~~単純なピーク検出~~
  - ビートに直接対応しないエネルギーピークが多い
- ビートという特定の認識対象の信号が入力音中にあるわけではない
- 各音の発音時刻を音響信号中から正確には求められない
  - 多数の楽器音が混在していて音源分離できない
- 複数のビートの解釈が可能
  - 一通りの解釈だけでは不十分
  - 正しい解釈に復帰することが必要



- 強拍・弱拍の判断や４分音符の判断が難しい
  - 音楽的知識を用いて判断することが必要

## □ 必要な処理

### ○ 複数の手がかりを抽出

- 様々な感度・周波数帯域で発音時刻を検出
- 主要なドラム音（バスドラム・スネアドラム）の発音時刻を検出　BD　SD

→ 計算量が多い

### ○ 複数の解釈の可能性を同時に追跡

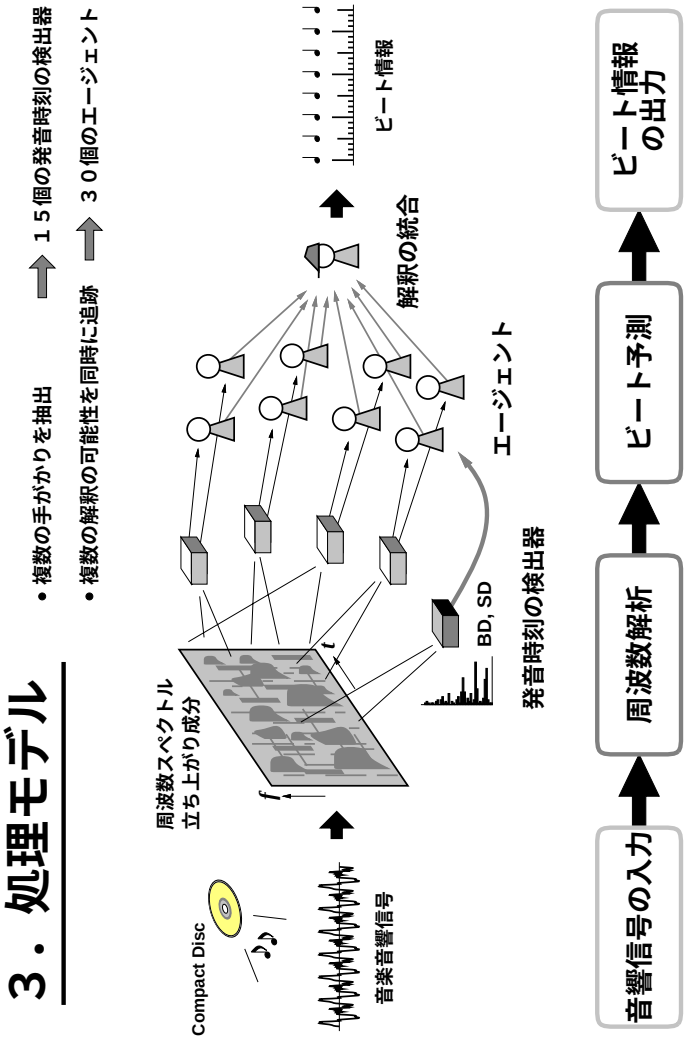
- 様々な解釈をする複数のエージェントが並列に次のビートを予測
- 各エージェントが解釈の確信度を自己評価
- 最も確信度の高い解釈を選択

→ 計算量が多い

**リアルタイムに実行するには並列処理が必要**

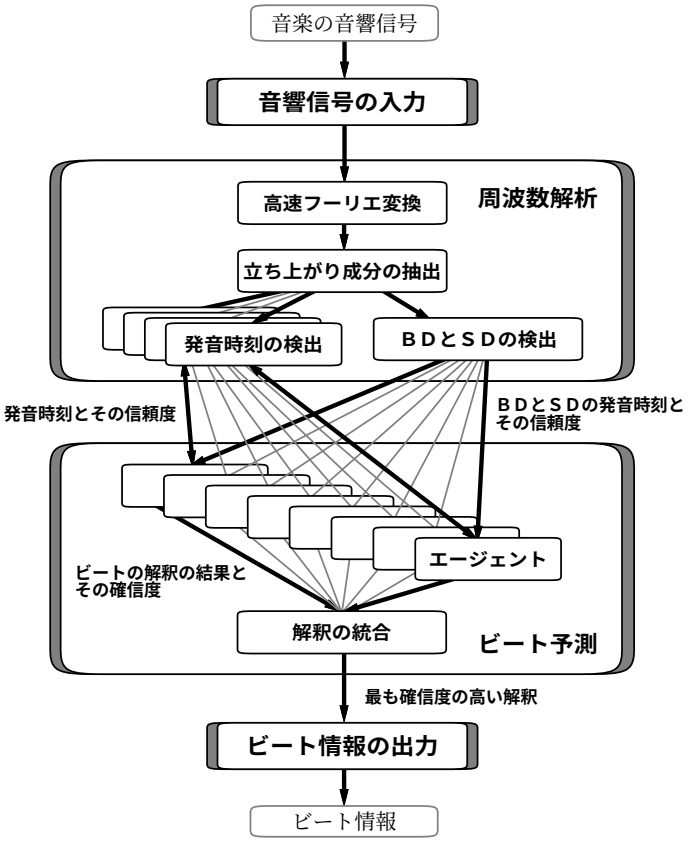


## 3．処理モデル



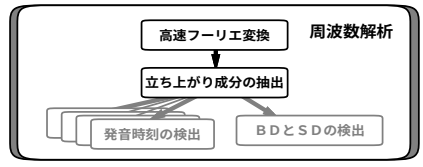



## □ 処理の流れ



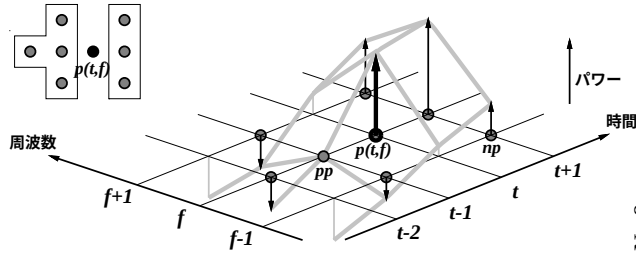



## □ 周波数解析

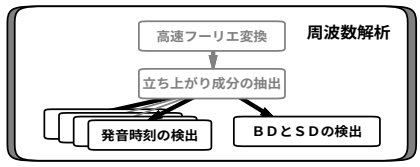



### ○ ＦＦＴ

### ○ 立ち上がり成分の抽出



## □ 周波数解析




### ○ 発音時刻の検出

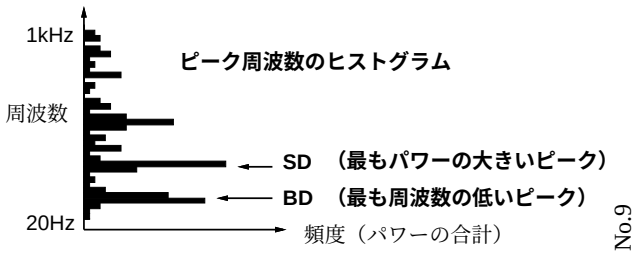
- １５個の発音時刻の検出器

様々な感度・周波数帯域で発音時刻を検出

### ○ ＢＤとＳＤの検出

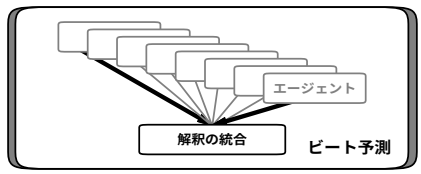
- １個のＢＤとＳＤ専用の検出器

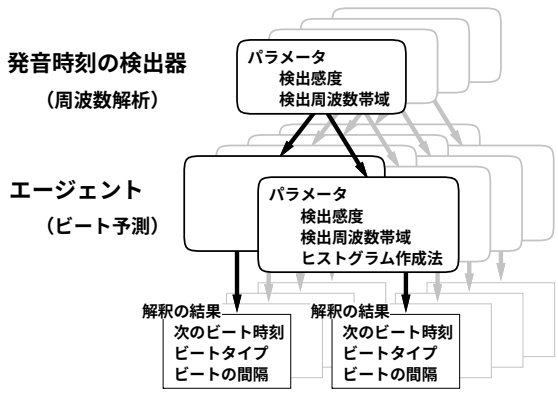
ＢＤとＳＤの特徴周波数を動的に獲得して検出






## □ ビート予測



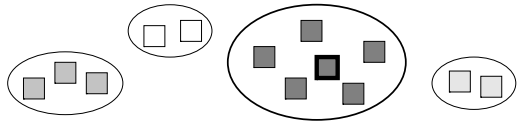

### ○ 解釈の統合

- 解釈の結果をグルーピング

ビート時刻・ビートの間隔

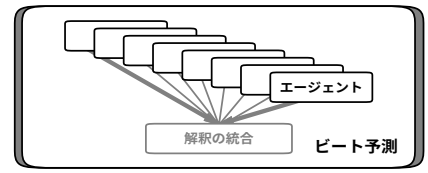
グループ内の全解釈の確信度を合計

- 最も確信度の高いグループ内の
  最も確信度の高い解釈を選択





## □ ビート予測



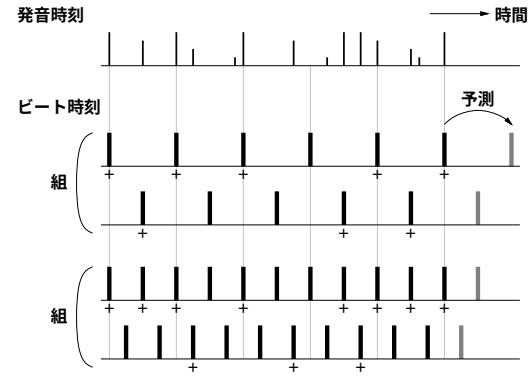

### ○ エージェント

- ３０個のエージェントが並列に予測

お互いに異なる戦略でビートを解釈

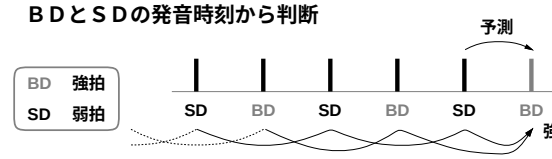
解釈の確信度を自己評価

- ２つずつ１５組に分けられる




- ビートタイプの判断

ＢＤとＳＤの発音時刻から判断



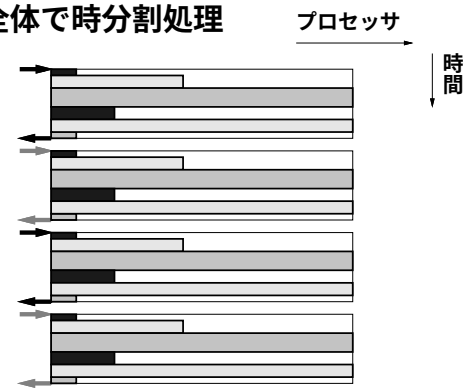



# ４．並列化手法

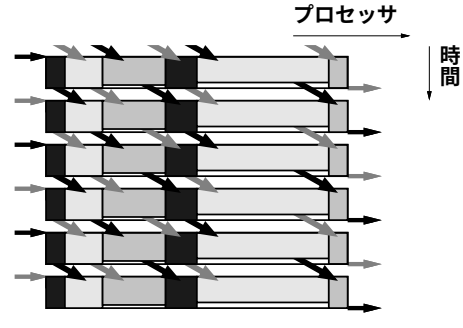
質の異なる多様な処理をリアルタイムに実行

並列化するには．．．

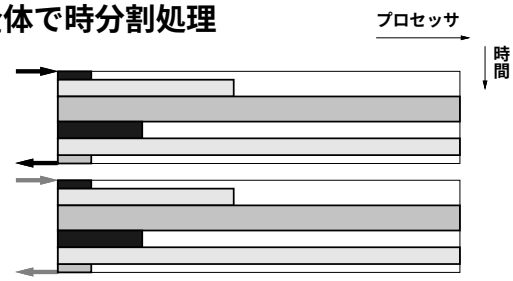
### ○ 計算機全体で時分割処理




### ○ 計算機全体でパイプライン処理



## × 計算機全体で時分割処理



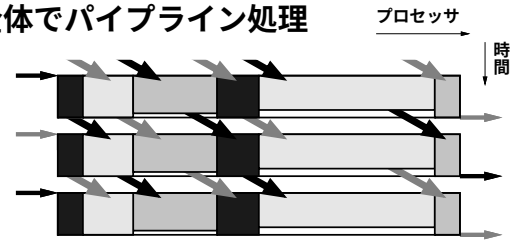

・並列度の低い処理　→　アイドル状態になるプロセッサ増
（ＢＤとＳＤの検出、解釈の統合）

・処理の切替えのオーバヘッド大

処理間の通信が終了するのを毎回待つ必要がある

通信時間のすべてが常に全体の実行時間に影響

## ◎ 計算機全体でパイプライン処理




・処理の並列度に応じてプロセッサを割り当てる
→　アイドル状態になるプロセッサ減

・常に割り当てられた同種の処理を実行
→　処理の切替えのオーバヘッド小

処理間の通信を一度におこなう

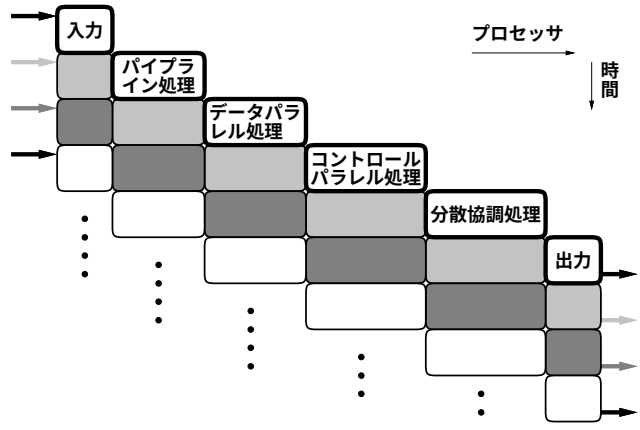
最も時間のかかる通信時間だけが全体の実行時間に影響



## □ 異なる並列処理方式を組み合わせたパイプライン処理

### ○ パイプライン中の各ステージにおいて処理の質に応じて異なる並列処理方式を適用

- パイプライン処理
- データパラレル処理
- コントロールパラレル処理
- 分散協調処理




### ○ 各ステージのプロセッサ数

総プロセッサ数（６４台）を考慮して決定

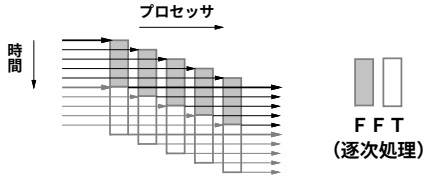


## □ 各処理ステージの並列処理方式


### ○ 周波数解析

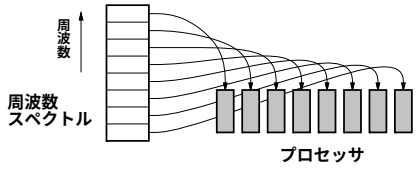
- **ＦＦＴ**　（逐次処理で 4.2ft ）　1ft = 11.61ms

**パイプライン処理**　**（５プロセッサ）**



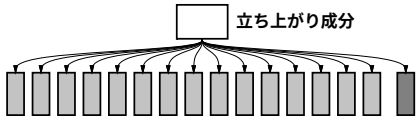

- **立ち上がり成分の抽出**　（ＤＯＡＬＬ型ループ）

**データパラレル処理**　**（８プロセッサ）**




- **発音時刻の検出器**　（１５個）
  **ＢＤとＳＤ専用の検出器**　（１個）

**コントロールパラレル処理**　**（１６プロセッサ）**



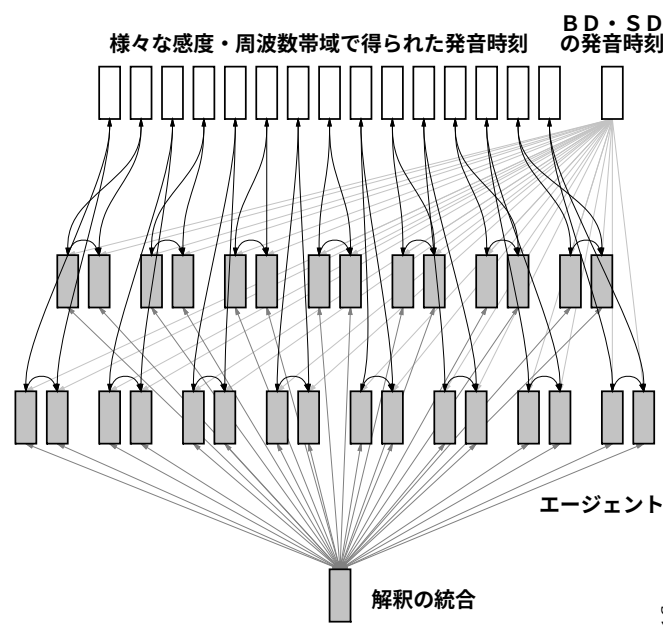



## □ 各処理ステージの並列処理方式

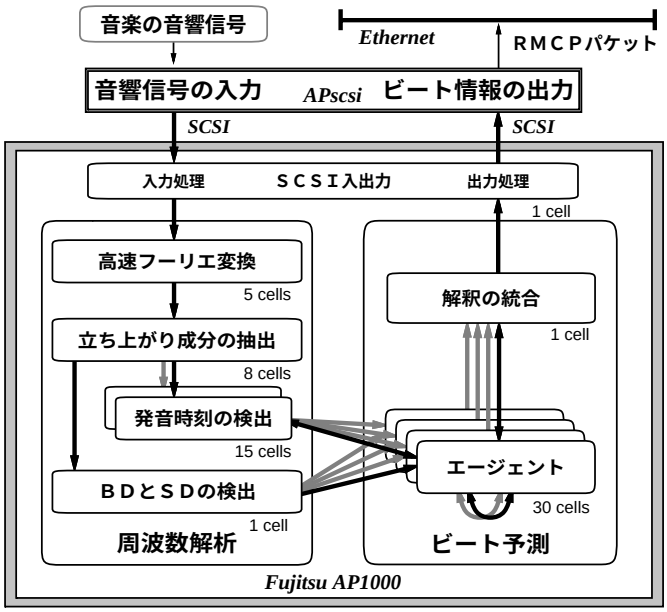

### ○ ビート予測

- **エージェント**　（３０個）
  **解釈の統合**　（１個）

**分散協調処理**　**（３１プロセッサ）**



# ５．ＡＰ１０００への実装




- ６４台のセル（要素プロセッサ）
- 各グループが計算機全体のパイプラインを構成

## ○ 入出力用計算機　*(APscsi)*

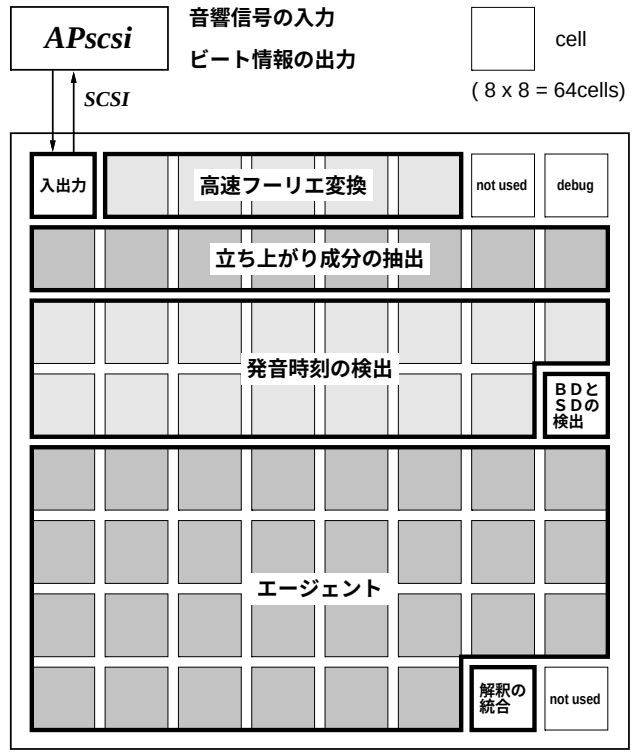
- Ａ／Ｄ変換

16bit / 22.05kHz　1block = 256 samples

ブロック化してＳＣＳＩ入出力セルへ転送



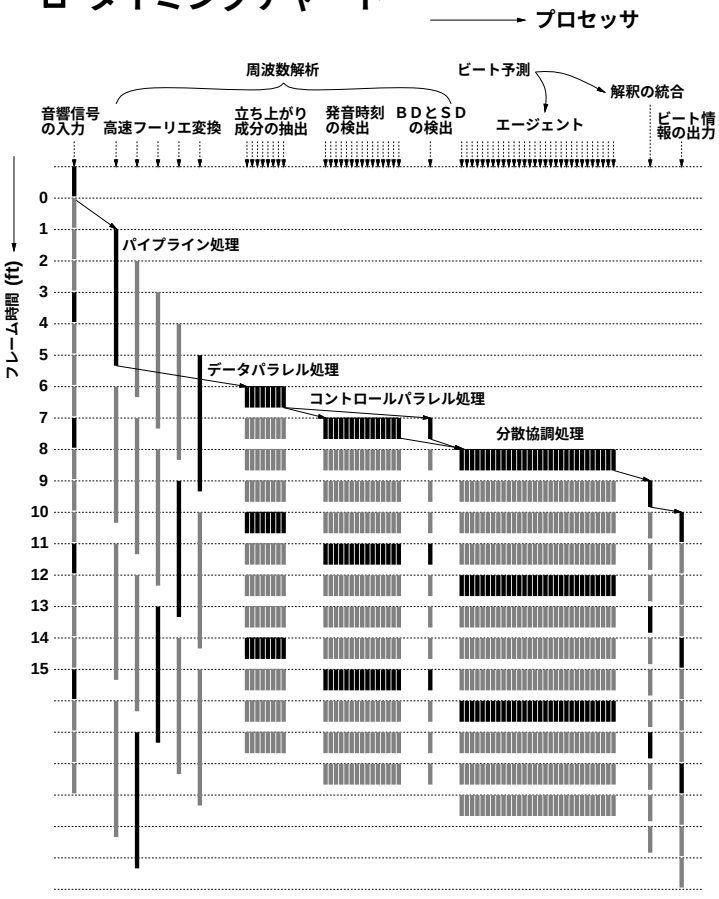
## ロ ＡＰ１０００セル割り当て




*AP1000*



## ロ タイミングチャート




**時間分解能：** 1 ft (frame-time) = 256 samples / 22.05kHz
= 11.61ms



## ロ 時間管理のための同期機構

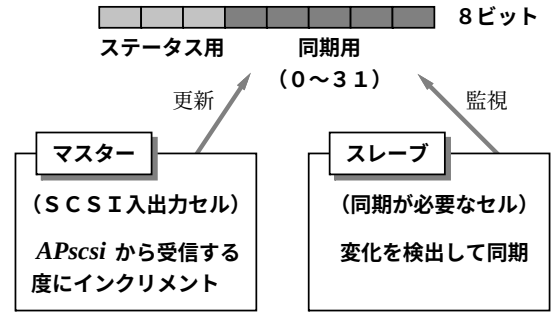
### • ＡＰ１０００内部の同期

- *APscsi* から1 ft ごとに転送されてくる音響信号の受信時刻に各セルの処理のタイミングを合わせる
- 同期が不要なグループは処理を続ける必要がある

~~Ｓ－ＮｅｔによるバリアＳ同期~~

### • ステータスハードウェアによる時間確認可能な同期機構

- 全セルからのステータスの論理和を各セルが参照




- 同期用ステータスの値が予想値より進んでいる
  → 処理が遅れていることがわかる

現在の実装では処理能力は計算負荷よりも十分に高い

## ロ ターンアラウンドタイム

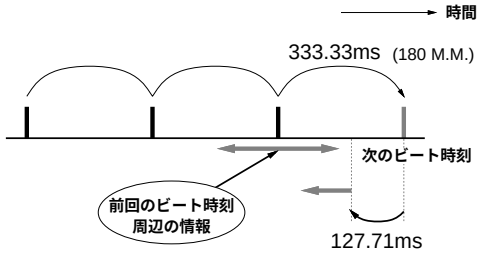
- **ＡＰ１０００全体のパイプラインを**
  **データが通過する時間**

11 ft = 127.71 ms

**（時間分解能：** 1 ft = 11.61 ms**）**

- 127.71ms**前の入力までが出力に反映される**

**過去のビートを認識して常に予測結果（次のビート時刻）**
**を出力する場合には問題にならない**




| テンポ | ４分音符の長さ |
|---|---|
| 70 M.M. | 857.1 ms |
| ≀ | ≀ |
| 180 M.M. | 333.3 ms |

- **ターンアラウンドタイムが短いほど**
  **より新しい情報を利用して予測ができ望ましい**



# ５．実験結果

### ○ 実験条件

- **ＣＤからサンプリングしたモノラルの音響信号を入力**
- **ＢＤが１・３拍目、ＳＤが２・４拍目で多く鳴るロックやポップス**
- **３０曲　（テンポ７８～１６８、２１アーティスト）**

### ○ 実験結果

- **２７曲で正しい結果**
- **曲の先頭ではビート時刻を認識できてもビートタイプは判定できない**
- **ドラムスが安定してから数小節後にビートタイプも正しく出力**

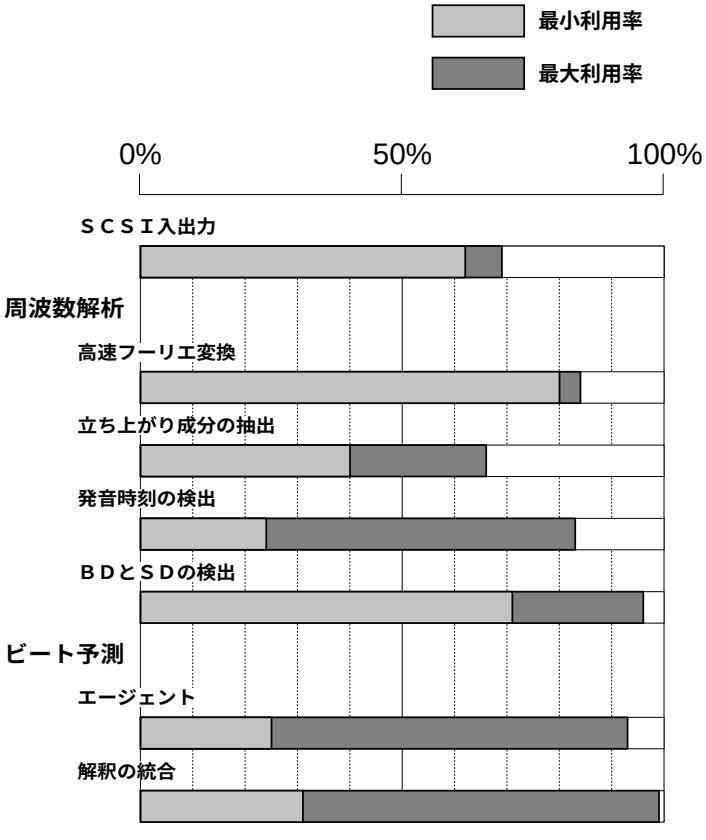
### ○ 考察

- **３曲の誤認識の原因**
  - **ビート時刻は正しく得られたがビートタイプを誤った**
  - **「Ｐｒｉｄｅ」「ラブストーリーは突然に」**
    **ＢＤとＳＤの特徴周波数が正しく獲得できなかったため**
  - **「Ａｒｃａｄｉａ」**
    **ドラムスの変則的なリズムが一時期続いたため**

➡ **実際の音響信号に対する本手法の有効性が確認された**



## ロ セル利用率






# ７．おわりに

## ロ まとめ

- **ドラムスを含む複数の楽器による音響信号に対し**
  **リアルタイムにビートを認識・予測**

- **複数の手がかりをリアルタイムに抽出**
  **ビートの様々な解釈の可能性を同時に追跡**

- **並列計算機全体でパイプライン処理**
  **処理の質に応じた４種類の並列処理方式を適用**

  **質の異なる様々な処理をリアルタイムに実行**

- **時間確認可能な同期機構**

  **同期が不要なプロセッサは処理を続行できる**

- **実際の音楽音響信号に対してリアルタイムに動作**

## ロ 今後の研究

- ビート予測のアルゴリズムの改良
- エージェント間の協調・統合の方式の改良
- 入力に対する制約を緩める
- より大局的な音楽構造の理解
- マルチメディアシステムへの応用