4S-4 動作履歴をグラフで表現するハードウェアの機能動作モデルについて

手嶋 茂晴, 長瀬 宏, 瀧川 光治
豊田中央研究所

1 はじめに

上位の論理設計では, ハードウェア機能記述言語は不可欠となってきている. 現在, 抽象的な並列アルゴリズムを記述できるようにハードウェア機能記述言語の拡張が進められているが, 多くの場合形式的意味が与えられていないために記述された動作にあいまいな部分が残る.

そこで本稿では, 操作的意味の立場からハードウェア機能記述言語に意味を与えるための機能動作モデルを提案する. 操作的意味とは, 言語を既に意味が明かになっている計算モデル (抽象機械) に対応づけることによって言語の意味を定義することである. ここでは, 提案する機能動作モデルが意味を定義する計算モデルとなる.

本稿では, まず機能動作モデルが取り扱うデータの構造 (動作履歴と呼ぶ) を定義する. 次に機能動作モデルを定める.

2 動作履歴

ハードウェア記述言語の意味を次のように考える. 「意味」とは, "入力パターン" を "動作終了時までの信号値の記録" (シミュレーション結果) に対応づけることである. ここで, 入力パターンや動作終了時までの信号値の記録など, 動作開始からある時刻までの信号値を記録したものを**動作履歴**と呼ぶ.

動作履歴を次のような有向グラフで表現する.

ノード: 機能的にひとまとまりになる信号値列.
時間幅を持つ区間を動作の最小単位とする. これを**最小動作**と呼ぶ.

枝: 時間の前後関係.
ノードaからノードbに向う枝は, ノードaに相当する最小動作の終了後にノードbに相当する最小動作bが開始されたことを示す.

ラベル: 値.
それぞれの最小動作に1つの値を割り当てる.

図1に示すタイミングチャートは図2のグラフで表現される.

動作履歴は次のような特徴を持つ.

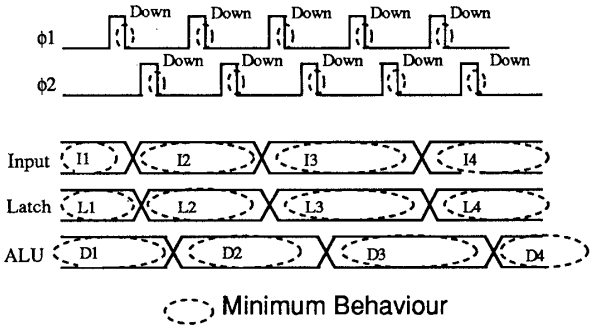

図 1: タイミングチャート

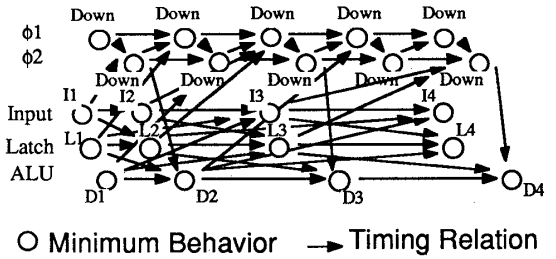

図 2: 動作履歴

- 様々な抽象度の記述に対応できる.
  何を最小動作に設定するかは抽象度によって自由に決めることができる.
- 暗示的な並列性を表現する.
  枝は逐次処理を表している. 枝をたどることのできない2つのノードは, それらに相当する最小動作が並列処理されることを表す. したがって, 逆に並列処理を明示することができない.

3 機能動作モデル

動作履歴はある時刻において, ハードウェアがそれまでどのように動作して来たかを表している. したがって, ここでの計算とは与えられた動作履歴 (つまり, 入力パターン) にノードと枝を追加して行き, 最終的な動作履歴を構成することである.

動作履歴にノードと枝を追加する抽象機械を考える. この抽象機械の動作は, 動作履歴にノードと枝を追加するための追加規則の集合で定義される. 追加規則は条件部と動作部から構成される. 追加規則の条件部は, 動作履歴の部分に対する照合条件を記述する. 動作部はどのようなノードと枝を追加するかを記述する. 抽象機械は

Computational Model on DAG for Operational Semantics of Behavioral Hardware Description Languages
Shigeharu Teshima, Hiroshi Nagase and Mitsuharu Takigawa
TOYOTA Cent. Res. & Develop. Labs.

条件部の条件に照合する動作履歴の全ての部分に対して,動作部で指定されたノードと枝を動作履歴に追加する(図3).

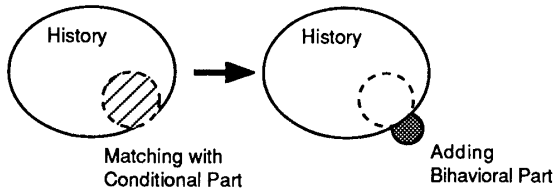

図 3: 追加規則

例 ペトリネット

図4に示すペトリネットを考える.「Placeに Tokenが存在すること」を“最小動作”とする.“最小動作の値”は, Tokenが存在した Place名 (p_1, p_2, p_3) とする.

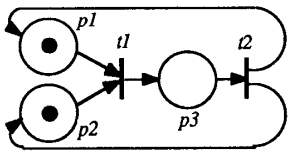

図 4: ペトリネット

追加規則は Trasitionに対応して, 以下の3つである.

1. **条件部:** p_1, p_2を値に持つ2つのノードの間を枝でたどることができない. (つまり, 同時に存在する.)

   実行部: 値p_3のノードを追加し, p_1, p_2を値に持つ2つのノードから,追加されたノードに向かって枝を張る.

2. **条件部:** p_3を値に持つノードが存在する.

   実行部: 値p_1のノードを追加し, p_3を値に持つノードから,追加されたノードに向かって枝を張る.

3. **条件部:** p_3を値に持つノードが存在する.

   実行部: 値p_2のノードを追加し, p_3を値に持つノードから,追加されたノードに向かって枝を張る.

追加規則1は Transition t_1に対応し, 追加規則2, 3は Trasition t_2に対応する.

動作履歴にノード, 枝を追加して行く様子を図5に示す.

今回提案する機能動作モデルは次のような特徴を持つ.

- 記述対象が動作する時間軸と抽象機械が動作する時間軸が別である.

  抽象機械は記述対象の動作する時間を操作の対象としている.記述対象の動作する時間をメタに扱っている.

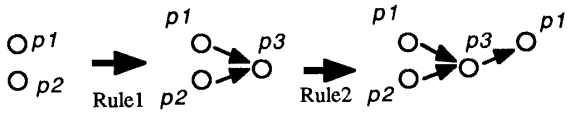

図 5: 動作履歴への追加過程

- ノードと枝の追加の順序に関係なく, 最終的に得られる動作履歴を一意に決めることができる.

  追加規則を制限することによって, 上の性質が保証される [2].

以上の特徴から抽象機械はそれ自身では並列計算の能力を持たなくても, ハードウェアの並列動作をあいまいなく定義することができる. したがって, ノードと枝の追加に基づく機能動作モデルによって, ハードウェア機能記述言語に操作的意味を与えることができる.

4 まとめ

ハードウェア機能記述言語に操作的意味を与えるための計算モデルを提案した.

従来のハードウェア機能記述言語は手続き的な計算モデルや有限状態機械モデルを意味づけの拠り所としている. いわゆるアルゴリズムレベルを記述する場合, 手続き的計算モデルを用いる. また, RTレベルの動作を記述する場合, 有限状態機械モデルを用いる. しかし, 手続き的計算モデルは本質的には, 有限状態機械と等価であるから, アルゴリズムレベルと言われている記述レベルは RTレベルの構文上の拡張に過ぎない.

有限状態機械モデルと本稿で提案する機能動作モデルとの最も大きな違いは動作履歴の表現形式にある. 本稿の能動作モデルでは, グラフによって, 時間関係(半順序)を表現するために, 絶対的な時間軸が存在しない. したがって, 並列動作を局所的な時間上に定義することができる. つまり, 1つの処理を, 他の処理との時間的な要素を考慮せず, 全く独立に記述することができる.

現在, この機能動作モデルの記述能力を検討するために, 抽象機械の追加規則を記述する言語の開発を進めている. この言語はハードウェア記述言語の意味定義の言語であると同時に, それ自身が最も基本的なハードウェア記述言語となる.

参考文献

[1] 石浦, 矢島: “ハードウェア記述言語の意味づけのためのハードウェアのモデルについて”, 情処第37回全国大会, pp. 1721-1722.

[2] 手嶋, 平石, 矢島: “イベントの2項関係に基づく並列システムの代数的仕様記述”, 信学論(D) Vol.J70-D No.1, pp. 19-29.