部分スキャンによる同期化可能な有限状態機械の合成について

On the Synthesis of Synchronizable Finite State Machines with Partial Scan

四浦 洋　井上智生　増澤利光　藤原秀雄
Hiroshi YOURA　Tomoo INOUE　Toshimitsu MASUZAWA　Hideo FUJIWARA

奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science
Nara Institute of Science and Technology

あらまし

本論文では、部分スキャンを用いて有限状態機械を同期化する問題について考察する。はじめに部分スキャンを用いた有限状態機械の同期化の原理を示し、通常入力とスキャン入力を組み合わせて有限状態機械を同期化する拡張同期化系列を提案する。次に最小個のスキャン可能な状態変数で有限状態機械を同期化可能とする問題、および、スキャン可能な状態変数の個数が与えられたとき最短の拡張同期化系列を求める問題について考察する。

Abstract

In this paper, we present an approach to synthesizing synchronizable finite state machines using partial scan. We propose an extended synchronizing sequence which consists of normal input-patterns and partial-scan input-patterns, and synchronizes an FSM. We consider two extended synchronizing problems; to obtain the minimum number of scannable state-valiables to synchronize an FSM, and to obtain the shortest extended synchronizing sequence and its state assignment for an FSM, given the number of scannable state-valiables.

1 はじめに

有限状態機械を初期状態に依存せず、ある特定の状態に遷移させることを、有限状態機械の同期化という。また、同期化のために入力される入力系列を同期化系列という。順序回路のテスト容易性として、対応する有限状態機械が短い同期化系列を有するという性質が挙げられる。一般に有限状態機械には同期化系列が存在しない場合があり、存在する場合でも系列長が非常に長くなる場合がある。同期化系列の長さが長くなればテスト系列が長くなり、テスト容易とはいえなくなる [1][2]。

N.Jiang ら [3] は有限状態機械に対して部分スキャンを行うことで、短い同期化系列を持つ順序回路を合成する方法を提案している。しかし、この方法ではスキャン操作は一回のみと仮定しているため、スキャン可能な状態変数の数や、その同期化系列の長さが必ずしも最小とは限らない。

本論文ではスキャン操作を複数回適用するものとして、通常入力とスキャン入力による拡張同期化系列を提案する。次に有限状態機械を最小個のスキャン可能な状態変数で拡張同期化する問題について考察する。さらにスキャン可能な状態変数の個数が与えられたとき最短の拡張同期化系列を求めるアルゴリズムを提案する。

2 有限状態機械の同期化

2.1 同期化系列による有限状態機械の同期化

有限状態機械の同期化、および、同期化系列は次のように定義される。

[定義 **1**][2] 同期化、同期化系列
有限状態機械 Mの初期状態に依存せず、ある特定の状態へ遷移させることを Mの同期化という。入力系列 Xを印加することにより、有限状態機械 Mを同期化することができるとき、この入力系列 Xを有限状態機械 Mの同期化系列 (synchronizing sequence) という。■

与えられた有限状態機械 (Finite State Machine; 以下 FSM と記す) が同期化系列を有するかどうかを調べたり、同期化系列が存在する場合にそれを求める問題は同期木 (図 2) を作ることにより解くことができる [2]。同期木において、各節内の状態の数を曖昧度 [2](ambiguity) という。図 2において、$\{s_0, s_1, s_3\}$ の曖昧度は 3 である。

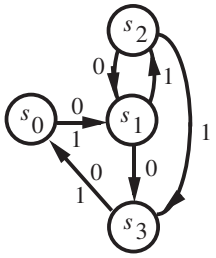

図 1: FSM M_1の状態遷移図

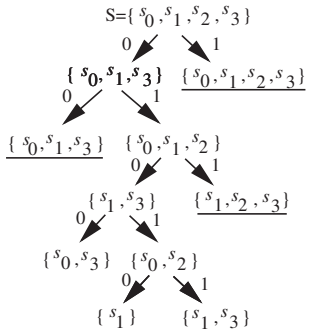

図 2: M_1の同期木

例 **1:** 図 1の FSM M_1の状態集合は $S = \{s_0, s_1, s_2, s_3\}$ である。M_1は S(曖昧度=4) から入力系列 01010 を印加することにより、$\{s_1\}$(曖昧度=1) となり、状態 s_1に同期化できる (図 2)。■

すなわち同期化系列とは FSM の状態集合を S、$|S| = n$ とすると、S(曖昧度=n) から単一の状態からなる集合 (曖昧度=1) に遷移させるような入力系列である。図 3の FSM M_2においては同期木は図 4 のようになり、同期化できない。次節ではこのような FSM を部分スキャンにより同期化する方法について述べる。

2.2 部分スキャンを用いた拡張同期化

本節ではまず部分スキャンによる同期化の原理について説明し、次に部分スキャンを用いて FSM を同期化する方法について説明する。

FSM の状態割当てに用いられる状態変数は、順序

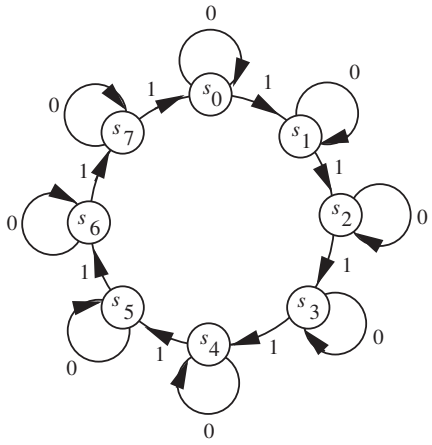

図 3: FSM M_2の状態遷移図

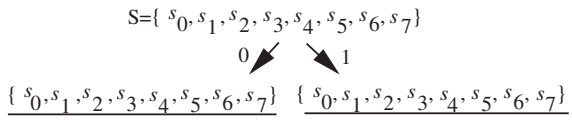

図 4: M_2の同期木

回路の記憶素子 (フリップフロップ) に対応する。部分スキャンとはいくつかのフリップフロップをスキャン操作により 0、1 どちらの値にでも設定できるようにする設計である。このようなフリップフロップに対応する状態変数をスキャン可能な状態変数という。また、リセット操作によりフリップフロップの値を 0、1 どちらか一方の値にのみ設定できるようにする設計もある。このようなフリップフロップに対応する状態変数をリセット可能な状態変数という。

m 個の状態変数 $y_0, y_1, \cdots, y_{m-1}$を持ち、そのうちの k個がスキャン可能な状態変数である FSM を考える。一般性を失うことなく、$y_0, y_1, \cdots, y_{k-1}$をスキャン可能な状態変数とし、$y_k, y_{k+1}, \cdots, y_{m-1}$をそれ以外の状態変数とすることができる。

FSM の状態集合を $S = \{s_t | 0 \leq t \leq 2^m - 1\}$ とする。スキャン可能な状態変数による Sの分割として、

$$\pi_s = \{T_0, T_1, \cdots, T_{2^k-1}\}$$

$$T_i = \{s_t | y_0(s_t)y_1(s_t)\cdots y_{k-1}(s_t) = i\}$$

を考える。ここで $y_0(s_t)y_1(s_t)\cdots y_{k-1}(s_t) = i$ とは、$y_0(s_t)y_1(s_t)\cdots y_{k-1}(s_t)$ を 2 進数とみたとき、その値が i と等しいことを表す。このときπ_sをスキャン分割といい、T_iをスキャン分割ブロックという。以下ではスキャン可能な状態変数 $y_0y_1\cdots y_{k-1}$を i に設定するスキャン操作をスキャン入力 i を行うという。状態集合 Sはスキャン入力 i により T_iに遷移する。

同時に、非スキャン可能な状態変数による状態集合 Sの分割として、

$$\pi_{ns} = \{P_0, P_1, \cdots, P_{2^{m-k}-1}\}$$

$$P_j = \{s_t | y_k(s_t)y_{k+1}(s_t)\cdots y_{m-1}(s_t) = j\}$$

を考える。π_{ns}を非スキャン分割といい、そのブロック P_jを非スキャン分割ブロックという。

π_sとπ_{ns}の積について、次式が成り立つ。

$$\pi_s \cdot \pi_{ns} = \{\{s_0\}, \{s_1\}, \cdots, \{s_{2^m-1}\}\} = \mathbf{0}$$

非スキャン分割ブロック P_jはスキャン入力 i により $s_t \in T_i \cap P_j$なる単一の状態 $\{s_t\}$ に遷移する。

表 1: FSM M_2の状態割り当て

	y_2	y_1	y_0
s_0	0	0	0
s_1	1	0	0
s_2	0	1	0
s_3	1	1	0
s_4	0	0	1
s_5	1	0	1
s_6	0	1	1
s_7	1	1	1

スキャン可能な状態変数:y_0

例 2: FSM M_2に対して表 1のように状態割当てを行う。スキャン可能な状態変数を y_0とすると、M_2においてスキャン分割ブロック、非スキャン分割ブロックは

$$T_0 = \{s_0, s_1, s_2, s_3\}, T_1 = \{s_4, s_5, s_6, s_7\},$$
$$P_0 = \{s_0, s_4\}, P_1 = \{s_1, s_5\},$$
$$P_2 = \{s_2, s_6\}, P_3 = \{s_4, s_7\}$$

である。例えばスキャン入力 0 により SはT_0に遷移する。またスキャン入力 0 により P_0は $\{s_0\}(s_0 \in T_0)$ に遷移する。 ■

Sの部分集合 Rに対するスキャン分割 $\{Q_0, Q_1, \cdots, Q_{2^k-1}\}$ を考える。スキャン操作 $y_0 \leftarrow i$ により Rは $Q_i(\subseteq T_i)$ に遷移する。

例 3: 表 1 のように状態割当てが行われた FSM M_2 において、R=$\{s_0, s_1, s_3, s_4, s_5\}$ (曖昧度 $= 5$) とす

ると、スキャン入力 0 により $Q_0 = \{s_0, s_2, s_4\} \subseteq T_0$ (曖昧度=3) に遷移し、スキャン入力 1 により $Q_1 = \{s_1, s_3, s_5\} \subseteq T_1$(曖昧度=3) に遷移する。このようにスキャン入力により曖昧度を減らし、状態を特定することができる。 ■

2.3 拡張同期化系列

通常入力とスキャン入力による同期化を特に拡張同期化と呼ぶ。以下に FSM の拡張同期化、および、拡張同期化系列を定義する。

[定義 2] 拡張同期化、拡張同期化系列
FSM Mの初期状態に依存せず、スキャン入力と通常入力によりある特定の状態に遷移させることを M の拡張同期化という。入力系列 Xを印加することにより、FSM Mを拡張同期化することができるとき、この入力系列 Xを Mの拡張同期化系列 (extended synchronizing sequence) という。 ■

例 4: 表 1は FSM M_2に状態割当てを行ったものである。M_2は通常入力だけからなる同期化系列を持たない。そこで拡張同期化系列を用いて拡張同期化する。スキャン可能な状態変数を y_0とする。M_2はスキャン入力 0 を行ったあと、通常入力 1 を 2 回行い、その後スキャン入力 0 を行ったあと、通常入力 1 を行い、さらにスキャン入力 0 を行うと状態 s_3に同期化される (図 5)。 ■

本論文では拡張同期化系列はスキャン入力を括弧で括り (0)11(0)1(0) のように書く。

拡張同期化系列にスキャン入力が連続して現れる場合、これらのスキャン入力を 1 回のスキャン入力に置き換えることにより、より短い拡張同期化系列が得られる。従って、以下で扱う拡張同期化系列には、スキャン入力が連続して現れることはないものとする。

2.4 拡張同期化問題

少ないハードウェアオーバヘッドでテスト容易化するためには、スキャン可能な状態変数の数を最小にすること、および、拡張同期化系列を最短にすることが重要である。これまでの例では、状態割当てが決まっている FSM を扱ってきたが、FSM に短い拡張同期化系列を持たせるためには、状態の割当て方が重要である。また、状態割当てが決まっている場合、拡張同期化可能にするためには、すべての状態変数をスキャン可能にしなければならない FSM が存在することを示すことができる。したがって、ここでは次の 3 つの拡張同期化問題について考察するが、いずれの問題も与えられる FSM は状態割当てが行なわれていないものとする。

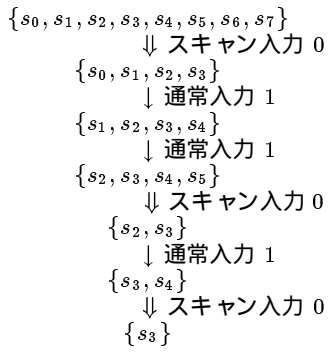

図 5: M_2の拡張同期化の流れ

[問題 1]
FSM Mが与えられたとき、最小個のスキャン可能な状態変数で Mを拡張同期化可能とするために必要な状態割当てを求める。 ■

[問題 2]
FSM Mとスキャン可能な状態変数の数 kが与えられたとき、最小個の状態付加により Mと等価な拡張同期化可能 FSM M'とその状態割当てを求める。 ■

[問題 3]
FSM Mとスキャン可能な状態変数の数 kが与えられたとき、最短の拡張同期化系列とその状態割当てを求める。 ■

3 スキャン可能な状態変数の最小化

本節では、前節で定義した問題 1、2 について考察する。

文献 [3] でも、スキャン操作は一回のみという仮定のもとで問題 2 を考察しており、FSM M が与えられたとき、等価な状態を追加することにより、1 個のリセット可能な状態変数を用いて長さ 2 の拡張同期化系列を持つ FSM M' を構成する方法を示している。この構成法では、M (状態数を n とする) の 1 つの入力に着目し、その入力によって i 個の状態が同一の状態に遷移する場合、$n - 2i$ 個の状態を追

加することにより M' を構成している。

本節では、以下の定理を示す。この定理は、スキャン操作を複数回行う場合について、同期化可能にするために必要なリセット可能な状態変数の数の上界 (問題 1)、および、k個のリセット可能な状態変数で同期化可能にするために付加する状態の数の上界 (問題 2) を与えている。

定理 **1** 状態数 n の状態機械の入力値 a に関する状態遷移図を $G(a)$ とする. また, $G(a)$ の弱連結成分の数を $m(a)$ と表す. このとき, 入力 a と状態変数のスキャンのみによる同期化に関して、以下が成り立つ.

(1) $G(a)$ に自己ループが存在しないとき

(a) $n \geq 4$, かつ, $G(a)$ が $n/2$ 個の長さ 2 のサイクルのみからなるとき ：

- n/2 が偶数の場合、1 個のスキャン可能な状態変数では同期化不可能。n/2 が気数の場合、1 個のリセット可能な状態変数では同期化不可能であり、1 個のスキャン可能な状態変数で同期化可能。また、いずれの場合も、2 個のリセット可能な状態変数で同期化可能。
- (等価な) 状態を 1 個追加すれば, 1 個のリセット可能な状態変数で同期化可能.

(b) その他の場合 ：
1 個のリセット可能な状態変数で同期化可能.

(2) $G(a)$ に自己ループが存在するとき.

- $(1-\frac{1}{2^{k-1}})n+1 < m(a) \leq (1-\frac{1}{2^k})n+1$ なら, $k-1$ 個のスキャン可能な状態変数では同期化不可能であり、k 個のリセット可能な状態変数で同期化可能.
- k個のスキャン可能な状態変数では同期化不可能な FSM に、状態を $\frac{2^k}{2^k-1}(m(a)-1)-n-1$ 個追加しても, k個のスキャン可能な状態変数では同期化不可能であり、(等価な) 状態を $\frac{2^k}{2^k-1}(m(a)-1)-n$ 個追加すれば, k個のリセット可能な状態変数で同期化可能。

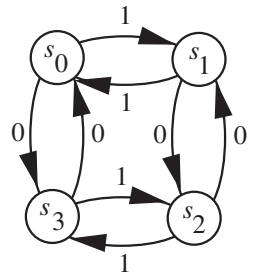

図 6: FSM M_3の状態遷移図

■

(注 **1**) 定理 1 において $k=1$ とすれば、(等価な) 状態を $2(m(a)-1)-n$ 個追加すれば, 1 個のリセット可能な状態変数で同期化可能であることが分かる。入力 a によって i 個の状態が同一の状態に遷移する場合、$m(a) \leq n-i+1$ となり、高々$n-2i$ 個の状態を追加することにより 1 個のリセット可能な状態変数で同期化可能であることが分かる。このことより、同期化可能にするために付加する状態数について、定理 1 は文献 [3] の結果の一般化である。

(注 **2**) 定理 1 では、2 種類以上の入力を利用した同期化を考慮していない。従って定理 1 では、同期化可能にするために必要なスキャン可能な状態変数の数の上界を与えているが、2 種類以上の入力を考慮した場合には、より少ないスキャン可能な状態変数で、同期化できる場合もある (1 個のスキャン可能な状態変数で同期化可能にするために付加する状態変数の数についても同様)。ただし、たとえ状態遷移図に自己ループを含まないような FSM であっても、2 種類以上の入力を利用しても 1 個のスキャン可能な状態変数では同期化できないものが存在する。図 6 はその一例である。従って、定理 1(1) で示す、同期化可能にするために使用するリセット可能な状態変数の数 (2 個) は、最悪時を考える限り、最小数である (1 個のスキャン可能な状態変数で同期化可能にするために追加する状態変数の数についても同様)。

定理 1 より、FSM が 1 個のリセット可能な状態変数で同期化可能なための十分条件は、ある入力 a について以下が成り立つことである。

1. $G(a)$ に自己ループが存在しないとき、$n \leq 3$, または, $G(a)$ が長さ 2 のサイクル以外の弱連結成分を含む。

2. $G(a)$ に自己ループが存在するとき、$m(a) \leq n/2+1$。

つまりこの手法において、1 個のリセット可能な状態変数では同期化できないのは、どの入力 a に関しても、$G(a)$ が $n/2$ 個の長さ 2 のサイクルからなるか、あるいは、$G(a)$ が $n/2+1$ 個より多い弱連結成分からなるときのみである。MCNC'89[4] のベンチマーク回路について確かめたところ、すべての回路がこの十分条件を満たし、1 個のリセット可能な状態変数で同期化可能であることが分かった。したがって、実際の回路では、この 1 個のリセット可能な状態変数で同期化できるための十分条件が成り立つと考えられる。

4 拡張同期化系列の最短化

本節では、問題 3 について考察する。与えられた FSM M とスキャン可能な状態変数の数 kに対して、最短の拡張同期化系列を求めるためには、(1) 系列を構成する通常入力、スキャン入力、(2) 状態割当て、(3) 無効状態の遷移、のすべての組合せの中から探索する必要がある。ここでは、効率良く短い拡張同期化系列を求めるために発見的手法を用いたアルゴリズムを提案する。

図 7に、発見的手法を用いた最短の拡張同期化系列を求めるアルゴリズムを示す。本アルゴリズムは、リセット状態として一つの状態を選択し、その状態に遷移する入力と状態の集合を求める逆順探索法 [3] (Reverse Order Search; ROS) に基づいている。以下に、アルゴリズムの流れを説明する。

1. リセット状態の選択 :全ての状態 $s \in S$について、その状態に通常入力により遷移する状態の集合を求め、最大の状態集合を持つ状態をリセット状態 s_rとする ($cand_reset()$)。現在の状態集合 S_cを $\{s_r\}$ とする。

2. 終了条件 :現在の状態集合 S_cが無効状態を含む全ての状態集合 $S \cup S_{inv}$であれば、それまでの入力系列を逆順に辿った系列を拡張同期化系列の解 ess として終了する。

3. 入力の選択 :現在の状態集合 S_cに遷移するような状態集合 S_pをすべての通常入力、スキャン入力について求め、状態集合 S_pが最大になる入力 i を求める ($select_input()$)。ここでスキャン入力による遷移を求めるため、以下の 3 つのステップを行う。

   (a) 状態割当ての選択 :スキャン入力による遷移は、状態集合 S_p内のすべての状態に対して状態割当てを行うことによって求め

```
min_ess(M, k)
    /* M: FSM, k: number of scannable
       state variables */
{
    S = valid_states(M);
    S_inv = invalid_states(M);
    I_n = normal_input_patterns(M);
    I_s = scan_input_patterns(k);
    l = 0;
    include = 1;
    s_r = cand_reset();
    S_c[l] = {s_r};
    while(S_c[l] ≠ S ∪ S_inv){
        i = select_input(S_c, ess, l);
        S_p = pred(S_c[l], i);
        ess[l] = i;
        l++;
        S_c[l] = S_p;
    }
    ess = reversle(ess);
    return ess;
}
select_input(S_c, ess, l)
{
    S_sc = S_c[l];
    for(i ∈ I_n){
        if(|pred(S_sc, i)| > |S_sp|){
            S_sp = pred(S_sc, i);
            i_s = i;
        }
    }
    if(l == 0 or ess[l-1] ∉ I_s){
        if(include){
            S_sc = S_c[l] ∪ S_inv;
            include = 0;
        }
        for(i ∈ I_s){
            state_assign(S_sc);
            if(|pred(S_sc, i)| > |S_sp|){
                Ssp = pred(S_sc, i);
                i_s = i;
            }
        }
    }
    return i_s;
}
```

図 7: 最短の拡張同期化系列を求めるアルゴリズム

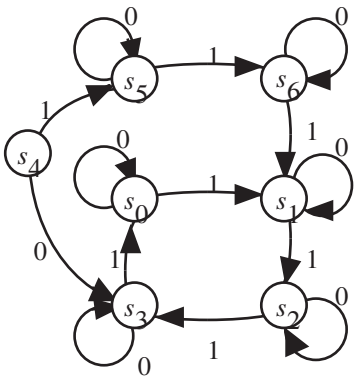

図 8: FSM M_4の状態遷移図

られる。ここでいう状態割当てとは、状態変数の値を決めることではなく、状態がどのスキャン分割ブロック、非スキャン分割ブロックに属するか決めることである。状態割当ては、そのスキャン入力の遷移を決定するのに必要な割当てのみが行われる。この状態割当ても、状態集合S_pが最大になるものが選ばれる。

(b) 状態割当てされる状態の選択 :S_c以外の状態で状態割当てされる状態は通常入力によりS_pに遷移する状態集合が最大になるように選択する。

(c) 無効状態の追加 :無効状態があれば、現在の状態集合S_cにそれを加えることで、スキャン入力によりS_cに遷移する状態集合S_pをより大きくする。ここでは、はじめに全ての無効状態をS_cに加える。また、S_cに無効状態を加えた時、その遷移はS_c内の任意の状態の遷移と同じにする。

4. 状態集合S_pを現在の状態集合S_cとして、ステップ 2 へ。

例 5: 図 8の$FSM\ M_3$、スキャン可能な状態変数の数 1 が与えられたとする。無効状態をs_7とする。はじめにスキャン分割ブロックT_0、T_1、非スキャン分割ブロックP_0、P_1、P_2、P_3 を用意する。

(1)s_1の通常入力 1 に対する状態集合S_pは$\{s_0, s_6\}$、s_3の通常入力 0 に対するS_pは$\{s_3, s_4\}$ であり、その他の状態の各通常入力に対するS_pの要素数は 0 または 1 である。s_1、s_3のうち、任意の状態を選択する。ここではs_0を選択する。$S_c = \{s_0\}$、$ess = NILL$。

(2)S_cの通常入力 0、1 に対する状態集合S_pはそれぞれ$\{s_1\}$、$\{s_0, s_6\}$ であり、各スキャン入力に対するS_pの要素数は 2 であるから、通常入力 1 を選択する。$S_c = \{s_0, s_6\}$p、$ess = 0$。

(3)S_cの通常入力 0、1 に対する状態集合S_pはそれぞれ$\{s_0, s_6\}, \{s_3, s_5\}$ である。ここでS_cに無効状態s_7を加える。状態割当てを$T_0 = \{s_0, s_6, s_7\}, T_1 = \{s_i, s_j, s_k\}$ $P_0 = \{s_0, s_i\}, P_1 = \{s_6, s_j, P_2 = \{s_7, s_k\}$のように行うと、スキャン入力 0 に対して、$S_p = \{s_0, s_6, s_7, s_i, s_j, s_k\}$ で最大となり、この状態割当てを選択する。s_i, s_j, s_kはそれぞれs_1, s_2, s_3となる。そしてスキャン入力 0 を選択する。$S_c = \{s_0, s_1, s_2, s_3, s_6, s_7\}$、$ess = 0(0)$。

(4) ここでは通常入力のみを考える。S_cの通常入力 0、1 に対する状態集合S_pはそれぞれ$\{s_0, s_1, s_2, s_3, s_4, s_6\}$, $\{s_0, s_1, s_2, s_3, s_5, s_6\}$, 通常入力 1 を選択する。$S_c = \{s_0, s_1, s_2, s_3, s_5, s_6\}$、$ess = 0(0)1$。

(5)T_0にs_4、T_1にs_5を加え、$P_3 = \{s_4, s_5\}$ とする状態割当てを行うと、スキャン入力 1 に対して、$S_p = S \cup S_{inv}$となる。$S_c = S \cup S_{inv}$、$ess = 0(0)1(1)$。

よって M_3の拡張同期化系列は $(1)1(0)0$ となる。

■

M_3の最短の同期化系列の系列長は 4 である。よって以上の発見的手法を用いた解は有効であると考えられる。

5 まとめ

本論文では、部分スキャンを用いて有限状態機械を同期化する問題について考察した。通常入力とスキャン入力を組み合わせて有限状態機械を同期化する拡張同期化系列を提案し、拡張同期化問題について考察した。その結果として、有限状態機械を拡張同期化するために必要なスキャン可能な状態変数の個数に関する定理、および、スキャン可能な状態変数の個数が与えられたとき、最短の拡張同期化系列を求めるアルゴリズムを示した。今後の課題として、ベンチマーク回路に対するアルゴリズムの有効性の評価、効率の良い発見的手法の開発などが挙げられる。

謝辞

本研究に関し、多くの貴重な意見をいただいた情報論理学講座の皆様方に感謝致します。

参考文献

[1] K. Cheng and V. Agrawal, "Initializability consideration in sequential machine synthesis," *IEEE Trans. on Computers*, vol. 41, pp. 374-379, Mar 1992.

[2] A. Miczo, "The Sequential ATPG: A Theoretical Limit," *Proc. Int. Test Conf*, pp.143-147, 1983.

[3] N. Jiang, R. M. Chou and K. K. Saluja, "Synthesizing Finite State Machines for Minimum Length Synchronizing Sequence Using Partial Scan. " *FTCS*, vol. 25, Jun. 1995.

[4] S. Yang, "Logic synthesis and optimization benchmarks user guide version 30," Microelectronics Center of North Carolina, Research Triangle Park, NC, Tech. Rep, Jan. 1991.

[5] I. Pomeranz and S. Reddy, "On the Role of Hardware Reset in Synchronous Sequential Circuit Test Generation," *IEEE Trans. on Computers*, vol. 43, pp. 1100-1105, Sep 1994.

[6] J. Rho, F. Somenzi, and C. Pixley, "Minimum Length Synchronizing Sequences of Finite State Machine." *Proc. Design Automation Conf*, pp. 463-468, 1993.

[7] H. Fujiwara, "Logic Testing and Design for Testability." *The Mit Press*, 1985.