Implications of Tracey's Theorem to Asynchronous Sequential Circuit Design

S. Gopalakrishnan, G. Kim and G. Maki
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - **Tracey's Theorem has long been recognized as essential in generating state assignments for asynchronous sequential circuits. This paper shows that Tracey's Theorem also has a significant impact in generating the design equations. Moreover, this theorem is important to the fundamental understanding of asynchronous sequential operation. The results of this work simplify asynchronous logic design. Moreover, detection of safe circuits is made easier.**

Introduction

Most digital controllers in use are synchronous sequential circuits. As the integration density of VLSI circuits increase, synchronous circuits face clock and power distribution problems. An asynchronous circuit on the other hand does not use a clock and hence the clock distribution overhead is eliminated. Further, the modules within an asynchronous circuit switch over a period of time instead of at a clock edge thereby reducing instantaneous current requirement and the complexity of the power distribution problem. Moreover, asynchronous circuits usually produce the fastest possible circuits as they are not limited in speed by the clock [1]. These advantages would project asynchronous controllers as a viable alternative. The major factor preventing the widespread use of asynchronous sequential circuits is that they involve a complex design procedure which must account for problems like critical races and hazards.

Tracey introduced a fundamental theorem that define the necessary and sufficient conditions for realizing state assignments [2]. This work shows the relationship between Tracey's state assignment procedure and design equations, that can impact the process of generating design equations.

This research was supported in part by the NASA Space Engineering Research Center Program under grant NAGW-1406 and the State of Idaho Grant SBOE 89-041. Dr. Gopalakrishnan is with the California Design Center of Hewlett Packard; Gui-hang Moon Kim is with The Mutual Life Insurance Company

y_1	y_2	y_3		I_1	I_2	I_3
1	1	1	A	B	C	A
1	0	0	B	B	D	C
0	1	0	C	E	C	C
0	0	0	D	E	D	C
0	0	1	E	E	E	A

Table 1: Asynchronous Sequential Flow Table with Tracey Assignment

y_1	y_2	y_3		I_p
0	0	0	A	A/1
0	1	0	B	A
1	1	0	C	C/0
1	0	0	D	D/1
0	1	1	E	F
1	0	1	F	F/0

Table 2: Partial flow table

Asynchronous Sequential Circuit Fundamentals

It is assumed that the asynchronous sequential circuits operate in the fundamental mode [1] and are encoded with single transition time (STT) state assignments [2]. This implies that once an input change occurs the circuit is allowed to reach a stable condition before any further input change is allowed. Moreover, all state variables that must change during a transition are excited to change simultaneously at the beginning of the transition. STT assignments allow for the fastest possible state transitions. STT state assignment procedures have been advanced by Tracey, Liu and Tan [2,3,4].

In a column of a flow table, all the k-1 unstable states which lead to the corresponding stable state, together with the stable state form a *k-set*. For example in Table 1, the k-sets under input I_1 are AB and CED. The set of states that a circuit can assume during a transition between the states of a transition pair is called the transition path. Liu's and Tan's state assignments partition the *k-sets* under an input column [3,4]. Tracey's state assignment procedure, on the other hand, partitions the transition pairs under a given input column [2].

Definition 1 *The two block partitions $\tau_1, \tau_2, \ldots \tau_n$, induced by the internal state variables $y_1, y_2, \ldots y_n$, respectively, are called the τ partitions of that assignment.*

For the state assignment given in Table 2, the τ partitions are as listed below.

$$\begin{aligned} \tau_1 &= \{CDF; ABE\} \\ \tau_2 &= \{BCE; ADF\} \\ \tau_3 &= \{EF; BCDA\} \end{aligned} \tag{1}$$

$\tau_1 = \{S_j \cdots ; S_k \cdots\}$ implies that the internal state Sj is coded '1' by the state variable y_1 and state S_k is coded '0'. Thus the first block of τ partitions are coded '1' by the corresponding internal state variables, and the second block of all τ partitions are coded '0' by the corresponding internal state variables.

Tracey's Theorem For STT operation in an asynchronous sequential circuit, all transition pairs of different *k-sets* under an input state must be partitioned by at least one state variable [2].

Tracey's theorem must be satisfied to eliminate critical races. Moreover, this theorem and the procedures used in this work [2] guarantees to produce all minimum variable state assignments. No other state assignment procedure can make such claims. Other assignment procedures may produce minimum variable assignments for some flow tables, but cannot guarantee to always produce the minimum variable assignment. Another feature of the Tracey assignment procedure is that it will generate all the STT assignments, not just the minimum variable assignments.

The design equations for an asynchronous sequential circuit can be formulated as follows:

$$\begin{array}{lcccccccccc} Y_1 & = & I_1(f_1^1) & + & I_2(f_1^2) & + \cdots + & I_p(f_1^p) & + \cdots + & I_m(f_1^m) \\ Y_2 & = & I_1(f_2^1) & + & I_2(f_2^2) & + \cdots + & I_p(f_2^p) & + \cdots + & I_m(f_2^m) \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ Y_i & = & I_1(f_i^1) & + & I_2(f_i^2) & + \cdots + & I_p(f_i^p) & + \cdots + & I_m(f_i^m) \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ Y_n & = & I_1(f_n^1) & + & I_2(f_n^2) & + \cdots + & I_p(f_n^p) & + \cdots + & I_m(f_n^m) \end{array} \tag{2}$$

Definition 2 *A next state partition η_i^p partitions those states where $Y_i = 1$ from those states where $Y_i = 0$, under the input I_p.*

The η partitions corresponding to the partial flow table given in Table 2 are given below.

$$\begin{aligned} \eta_1^p &= \{CDEF; AB\} \\ \eta_2^p &= \{C; ABDEF\} \\ \eta_3^p &= \{EF; ABCD\} \end{aligned} \tag{3}$$

Definition 3 *An f_i^p is a sum of products expression of internal state variables that partition the states of η_i^p.*

The f_i^p expressions corresponding to the next state partitions of the partial flow table shown in Table 2 are given below.

$$\begin{aligned} f_1^p &= y_1' y_3' \\ f_2^p &= y_1 y_2 \\ f_3^p &= y_3 \end{aligned} \tag{4}$$

Definition 4 *An internal state variable that partitions the states of one transition pair from the states of another is called a partitioning variable.*

It will be shown that the partitioning variables play an important role in the design equations.

Tracey Partitioning Variables

The relationship between the Tracey partitioning variables and the design equations has not been presented before. Consider the following development. Let S_i and S_j be states of a transition pair and S be the set of all states other than S_i and S_j.

Lemma 1

$$\begin{aligned} \textit{If}\quad \tau_1 &= S_iS_j; S_1 \\ \tau_2 &= S_iS_j; S_2 \\ &\vdots \\ \tau_k &= S_iS_j; S_k \end{aligned}$$

where $S_1 \cup S_2 \cup \ldots \cup S_k = S$, *then* $S_iS_j; S$ *can be covered by the product term* $y_1^* y_2^* \ldots y_k^*$, *where* $y_i^* = y_i$ *or* y_i'.

Proof: If states S_iS_j are in the same block of each τ_i in the set of τ-partitions, then the intersection of partitions τ_i (or τ_i') will produce partition S_iS_j ; S. The product expression that covers S_iS_j ; S is $y_1^* y_2^* \ldots y_k^*$.

Lemma 2

$$\begin{aligned} \textit{If}\quad \{\eta_1; S\} &\quad \textit{is covered by product term } P_1 \\ \{\eta_2; S\} &\quad \textit{is covered by product term } P_2 \\ &\vdots \\ \{\eta_m; S\} &\quad \textit{is covered by product term } P_m \end{aligned}$$

then the partition $\{\eta_1\eta_2 \ldots \eta_m; S\}$ *can be covered by a sum of products expression* $P_1 + P_2 + \ldots + P_m$.

Proof: The proof follows directly from fundamental partition algebra.

Theorem 1 *The* f_i^p *terms of Eq. (1) are a function of the partitioning variables of input* I_p.

Proof: Consider a transition path T_0 where $Y_i = 1$ in T_0 and $Y_i = 0$ in the other transition paths T_i in I_p. The next state partition η_i^p can be expressed as T_0;S, where S contains all other transition pairs from different k-sets than T_0. The goal is to find an expression that partitions T_0 from S. From Tracey's Theorem [2] there exists a set of partitioning variables τ_i that partition T_0 from all other T_j. Therefore from Lemma 1, there exists an expression for Y_i covering T_0 that consists only of the partitioning variables.

In general, $Y_i = 1$ in several transition paths under a given input. A portion of the next state partition η_i for those transition paths where $Y_i = 1$ can be depicted as $\eta_i^p = T_k$; S. However, each transition path T_k is partitioned by partitioning variables and can be expressed as a product of the state variables that partition the transition path where $Y_i = 1$ from those where $Y_i = 0$. Therefore by Lemma 2, each f_i^p term can be expressed as a sum of products of the partitioning variables from $I + p$. **QED**

The following theorem yields a surprising basic result which shows that partitioning variables do not change state during circuit transitions.

Theorem 2 *The partitioning variables that partition a transition path T_0 from all other transition paths do not change state while the circuit proceeds from unstable to stable state of T_0.*

Proof: Since both unstable and stable states of a transition pair of T_0 are coded the same by the partitioning variables, these partitioning variables are not excited as the circuit transitions between the states of T_0. **QED**

$y_1y_2y_3y_4y_5$		I_1	I_2	I_3
1 1 1 1 1	A	**A**	**D**	**H**
1 1 1 1 0	B	**A**	**B**	**C**
1 0 0 1 0	C	**C**	**B**	**C**
1 0 1 0 1	D	**C**	**D**	**F**
0 1 1 0 0	E	**E**	**D**	**E**
0 1 0 0 1	F	**E**	**F**	**F**
0 0 0 0 0	G	**G**	**F**	**E**
0 0 0 1 1	H	**G**	**H**	**H**

Table 3: Machine B

To illustrate Theorem 1 and to show its impact on classical logic design, consider the flow table shown in Table 3. The state assignment partitions all the transition paths and is a Tracey assignment. The state variables $\{y_1, y_2\}$, $\{y_3, y_4, y_5\}$ and $\{y_4, y_5\}$ are the partitioning variables that partition the transition paths under I_1, I_2, and I_3 respectively. From Theorem 1, the expression representing each next state variable is a function only of the following partitioning variables under each input:

I_1 y_1, y_2
I_2 y_3, y_4, y_5
I_3 y_4, y_5

I_1

Y2 \ Y1	0	1
0	GH 00000	CD 10010
1	EF 01100	AB 11111

I_3

Y5 \ Y4	0	1
0	EG 01100	BC 10010
1	DF 01001	AH 00011

I_2

Y5 \ Y3Y4	00	01	11	10
0	G 01001	C 11110	B 11110	F 01001
1	E 10101	H 00011	A 10110	D 10101

Figure 1: State Tables for Each Input

The state tables for each input need be constructed with only the partitioning variables associated with each input state. The significance of this example is that even though a 5 variable state assignment is needed in the traditional synthesis process, nothing larger than a 3 variable k-map need be constructed on a per input basis as shown in Fig. 1. The following design equations are obtained easily from Fig. 1.

$$
\begin{aligned}
Y_1 &= y_1 I_1 + (y_4 y_5' + y_3 y_5 + y_4' y_5) I_2 + y_4 y_5' I_3 \\
Y_2 &= y_2 I_1 + y_5' I_2 + y_4' I_3 \\
Y_3 &= y_2 I_1 + (y_4 y_5' + y_3 y_5 + y_4' y_5') I_2 + y_4' y_5' I_3 \\
Y_4 &= y_1 I_1 + (y_4 y_5' + y_3' y_4) I_2 + y_4 I_3 \\
Y_5 &= y_1 y_2 I_1 + (y_4' + y_5) I_2 + y_5 I_3
\end{aligned}
\tag{5}
$$

One could easily come to the conclusion that satisfying Tracey's conditions are sufficient to realize STT state assignments. This notion is not always true as depicted from an example taken from [5]. Shown in Fig. 4 is the flow table and the design equations follow. The state assignment satisfies Tracey's theorem.

$(y_4\ y_5\ y_1\ \cdots y_3)$	I_0	I_1	I_2	I_3
(1 1 0 0 0) A	B	A	A	A
(1 0 0 0 1) B	B	A	–	C
(0 1 0 1 1) C	D	A	G	C
(1 0 0 1 0) D	D	A	–	A
(0 1 1 0 1) E	F	E	A	E
(0 0 1 1 1) F	F	G	–	E
(0 1 1 1 0) G	H	G	G	E
(1 0 1 0 0) H	H	E	–	E

Table 4: Flow Table with Modified State Assignment for Fail Safe Realization

$$
\begin{aligned}
Y_1 = {} & y_5y_2y_3I_3 + y_5y_1y_3I_3' + y_1y_2y_3 \\
& + y_5y_1y_2 + y_4y_1 \\
Y_2 = {} & y_4y_3I_2 + y_5y_2y_3I_1' + y_4y_1I_0 + y_5y_1y_3I_0 + y_1y_2y_3I_2' \\
& + y_5y_1y_2(I_1 + I_3) \\
Y_3 = {} & y_4y_5I_0 + y_5y_3I_1' + y_5y_2y_3I_2 + y_5y_1y_3I_3' + y_1y_2y_3I_1' \\
& + y_5y_1y_2I_2 + y_4y_1I_0' \\
Y_4 = {} & y_4y_5 + y_4y_3(I_0 + I_1) + y_5y_2y_3(I_0 + I_1) + y_4y_2 + y_5y_1y_3I_3 \\
& + y_5y_1y_2I_0 + y_4y_1I_0 \\
Y_5 = {} & y_4y_5I_0' + y_4y_3I_0' + y_5y_3I_0' + y_4y_2I_0' + y_5y_1y_3I_0' \\
& + y_1y_2y_3I_0' + y_5y_1y_2I_0' + y_4y_1I_0'
\end{aligned}
\tag{6}
$$

The partition variables for each input are:

I_1 y_1, y_2
I_2 y_3, y_4, y_5
I_3 y_4, y_5
I_4 y_4, y_5

Notice that the state variables used in the design equations are not a function of the partitioning variables. An analysis of the state tables shows that this design is replete with critical races simply because the design equations were improperly generated.

Safe Sequential Circuits

The above results can be applied to analyze the safeness of sequential circuits. Conventional asynchronous sequential circuit design procedures generate circuits that implement the flow table as per specification; under normal conditions of operation it is assumed that the circuit assumes only those states shown in the flow table. However, under abnormal conditions, like power fluctuations or noise spikes, the circuit can enter unspecified states. This condition may go undetected and may even be impossible to correct without the application of a master reset. Further, in some circuits there may be no reset signal

available for resetting the circuit to a known stable state. Therefore, it is important that the designer be aware of the existence of such unspecified stable states in producing a safe design. A safe circuit is essentially a circuit in which it is always possible to assume specified stable states and the circuit is never allowed to remain indefinitely in unspecified states.

Unspecified stable states are formed when the design equations unintentionally specify the state as stable. The first study on safe circuit design was presented by Wickersham and Maki [6]. In this paper, a procedure for safe circuit detection was introduced. They also introduce a safe design procedure which may require more than the minimum number of state variables.

Safeness Analysis

Definition 5 *Two internal states of a sequential circuit are in the same equivalence class under I_p, if they have the same next state entry.*

Definition 6 *A lock-up state under an input I_p, is any stable state under I_p that is not a member of a transition path under that input.*

Definition 7 *A lock-up state and the states of the flow table that have a lock-up state as a next state entry form a lock-up equivalence class.*

When a circuit assumes a lock-up state, there are several ways to attempt force the circuit back to the specified states of the flow table. One of the schemes is to apply an input or a sequence of inputs so that the circuit transitions through a set of unspecified stable states until it finally reaches a specified stable state. However, in some cases this may not be possible. Another approach is to turn the power off and on and hope that the circuit would come up in a specified stable state. Once a specified state is reached, further input changes keep the circuit in its specified transition paths, until an abnormal condition forces it out of the specified transition path.

Definition 8 *A critical situation exists if there are states from which it is impossible to re-enter a specified transition path by simply changing inputs.*

Definition 9 *A circuit is* **safe** *if a critical situation does not exist.*

This implies that a safe circuit can be designed by making sure that there is at least one input column under which there are no lock-up states. If the circuit ever enters a lock-up state, the input with no lock-up states can be applied to force the circuit in to one of the transition paths of the circuit. However, for some situations it may be desired to ensure that all the inputs are lock-up free. The safeness analysis procedure is a modified version of the procedure outline by Wickersham and Maki [6], the difference being that only the partitioning variables under each input are used for the analysis instead of all the state variables being used. Procedure 1 is based on the following theorems.

Theorem 3 *Equivalence classes under an input state can be represented as functions of the partitioning variables under that input state.*

Proof: Equivalence classes are specified by the next state entries under an input. From Theorem 1, the next state equations are a function only of the partitioning variables. Therefore equivalence classes are a function only of the partitioning variables. QED.

Theorem 4 *To identify the stable states under an input state, only the present state and next state value of the partitioning variables under that input need be checked.*

Proof: For any input state, product expressions of the partitioning variables under that input state uniquely represent the stable state and the associated equivalence class. If the circuit is in one of the stable equivalence classes then it assumes one of these unique product expressions. To check whether a circuit is in a stable state under an input state, only the present state and next state values of the partitioning variables under that input state need be checked. QED.

Procedure 1 *New safeness analysis procedure.*

Step 1: Map the design equations on to a K-map. The K-map for each input need contain only the partitioning variables. A stable state occurs whenever present and next state are equal. Only the f_i^p terms for the partitioning variables under that particular input need be considered in this step. This can be repeated for all the inputs.

Step 2: Identify all the lock-up states. Lock-states are those stable states that are not specified in the flow table.

Step 3: If lock-up states are found then determine if a critical situation exists. This can be accomplished by determining which equivalence class the circuit enters under other inputs from the lock-up equivalence class. If there exists a lock-up equivalence class which transitions only to other lock-up equivalence classes and specified equivalence classes cannot be entered, then a critical situation exists. If a critical situation exists then the circuit is not safe.

Example 1 *Consider the flow table shown in Table 2, the state assignment used is the Tan assignment.*

The design equations for implementing this circuit is given in Equation 7.

$$\begin{aligned}
Y_1 &= I_1(y_1) + I_2(y_3) + I_3(y_5) \\
Y_2 &= I_1(y_2) + I_2(y_4) + I_3(y_6) \\
Y_3 &= I_1(y_1) + I_2(y_3) + I_3(y_6) \\
Y_4 &= I_1(y_2) + I_2(y_4) + I_3(y_5) \\
Y_5 &= I_1(y_2) + I_2(y_3) + I_3(y_5) \\
Y_6 &= I_1(y_1) + I_2(y_4) + I_3(y_6)
\end{aligned} \tag{7}$$

y_1	y_2	y_3	y_4	y_5	y_6		I_1	I_2	I_3
1	0	1	0	0	1	A	A	B	F
1	0	1	0	1	0	B	A	B	C
1	0	0	1	1	0	C	A	E	C
0	1	0	1	1	0	D	D	E	C
0	1	0	1	0	1	E	D	E	F
0	1	1	0	0	1	F	D	B	F

Table 5: Flow table for Safe Design

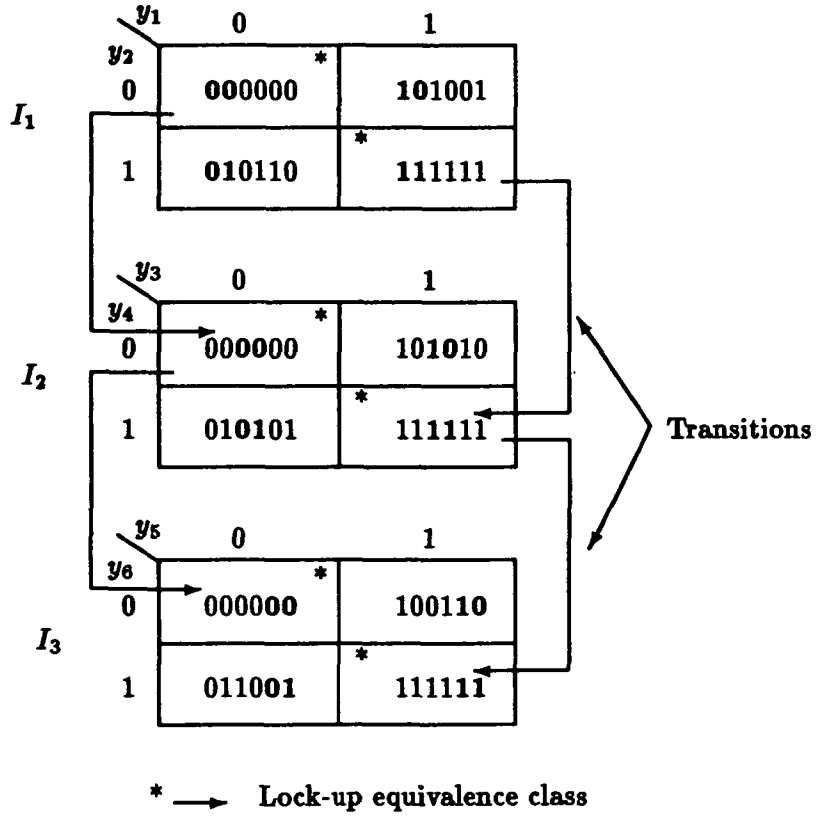

Figure 2: Safeness analysis

The partitioning variables under input I_1 are y_1, y_2, input I_2 are y_3, y_4 and input I_3 are y_5, y_6. The safeness analysis for this implementation is shown in Fig. 2. It can be seen that there are two lock-up states per input namely, 000000 and 111111. It can also seen that these lock-up states lead to a critical situation since once any of these states is assumed the circuit cannot re-enter a proper transition path by change of inputs.

Summary

The relationship between Tracey's fundamental state assignment theorem and the resulting design equations has been shown. The state variables that are partitioning variables to prevent critical races must appear in the design equations to insure STT operation. Classical design can be accomplished easier as a result of this work because k-maps with fewer state variables are required in representing the state tables. Showing state tables with only the partitioning variables allows easy analysis of asynchronous sequential circuits.

References

[1] S. Unger, *Asynchronous Sequential Switching Circuits*, New York, NY, Wiley-Interscience, 1969.

[2] J. Tracey, "Internal State Assignments for Asynchronous Sequential Machines", IEEE Transactions on Electronic Computers, Vol. EC-15, pp. 551-560, Aug. 1966.

[3] C. Liu, "A State Variable Assignment Method for Asynchronous Sequential Switching Circuits," J. ACM, Vol 10, pp. 209-216, Apr. 1963.

[4] C. Tan, "State Assignments for Asynchronous Sequential Machines", IEEE Transactions on Computers, Vol. C-20, No. 4, pp. 382-391, April 1971.

[5] S. Das and Y. H. Chuang, "A Unified Approach to the Realization of Fail-Safe Sequential Machines," IEEE Fault Tolerant Computing Symposium, pp. 3-2, 3-6, June, 1974.

[6] R. Wickersham and G. Maki, "Safe Asynchronous Sequential Circuits", IEEE Transactions on Computers, Vol. C-23, No. 5. May 1974.

Acknowledgement– **The professional children and grandchildren of Dr. James H. Tracey would like to acknowledge his contribution to the theory of circuit design. Moreover, we would like to express our appreciation for his personal encouragement in our lives. We regard him as a dear friend and sincerely admire the dynamic enthusiasm expressed through his life.**