An Algebraic Formulation of Seitz's Weak Conditions for Self Timed Circuits

S.Purushothaman

P.A.Subrahmanyam

Department of Computer Science
University of Utah
Salt Lake City, Utah 84112

UTEC–82–066
October 1982

Abstract

Two fairly intuitive conditions are given that serve to algebraically characterize Seitz's "weak conditions" for self timed circuits. It is shown that these two conditions embody the 12 temporal logic conditions (developed by Owicki and Malachi) which are intended to express both the weak conditions as well as certain liveness properties that self timed circuits need to satisfy.

This research was sponsored by Defense Advanced Research Projects Agency, US Department of Defense, Contract No.MDA903–81–C–0414.

1. Introduction

The current trends in VLSI design research point to the development of a theory for VLSI circuits quite analogous to the development of theories for program design and programming languages. The need for (and the advantages of) a rigorous basis that supports a broad spectrum of the VLSI design process is by now well recognized and has been widely argued for.

A uniform algebraic framework that supports abstract (functional), architectural, and electrical levels of abstraction, and that is, in particular, intended to aid in the mechanical development of designs proceeding from high level specifications, has been introduced in [4]. In recognition of the increasing importance of self timed implementations, we detail in this note an algebraic formulation of Seitz's weak conditions for such circuits. Our prime motivation here is to recast the work on Seitz's conditions based on temporal logic [1], so as to maintain the same algebraic framework at all the three levels of the hierarchy mentioned above. A by–product of this effort is that we have a fairly succinct characterization (in terms of two very intuitive and easy to state conditions) of the 12 conditions enumerated by Owicki and Malachi in [1].

In section 2 we review some basic definitions that are used in section 3 to algebraically formulate the "weak conditions". Finally, in section 4 we show how the temporal logic conditions proposed in [1] are embodied by the algebraic formulation.

2. A Model for VLSI Circuits. Some Preliminary Definitions

We adopt the prevalent notion that an electrical circuit is a network of processing elements / combinatorial modules (henceforth called modules) having a set of input and output ports. The modules perform computations and can communicate with other modules or with the external world through their ports. In this model, the function of a module can be captured by specifying the data values obtained at the output ports as a (mathematical) function of the values provided at the input ports. The functionality of the module (i.e., its behavior over a period of time,) can be captured by stating certain conditions that must hold on the sequence of input–output vectors.

We assume that the data on the I / O lines is an element of the set $D = \{0, 1, \perp\}$ where $\perp \leq 0$ and $\perp \leq 1$. We also assume that once an input line becomes defined, it retains the same value until all the outputs become defined. This implies that any transition from the value 0 to 1 or vice–versa should pass through the value $\perp$.

A function f is a mapping from a domain A to a domain B, i.e. f: A –> B. We further presuppose an partial ordering $\leq$ on the elements of A and B.

A function is said to be *total* if all the elements of A are mapped by the function f to some element of B, otherwise the function is said to be *partial*. It is useful to define the "natural–extension" of a partial function as being the padded function that evaluates to $\perp$ whenever any of its arguments is $\perp$. That is, a function f is said to be naturally–extended if

if $\exists\ a_j : a_j \varepsilon (a_1,...,a_n)$ and $a_j = \perp$ then $f(a_1,...,a_n) = \perp$.

Given a function f: A –> B, we say that it is monotonic if it "preserves" order, i.e. if it satisfies the condition,

$$(\forall\ a_1, a_2 \varepsilon\ A\ )\ a_1 \leq a_2 \supset\ f(a_1) \leq f(a_2)$$

A set $D = \{a_1,...a_n\} \subseteq A$, is said to be a *chain* if $a_1 \leq a_2 \leq ... \leq a_n$.

A least upper bound ($\bigsqcup$) of two elements a_1, a_2 belonging to a set A, is an element $b \in A$, such that the following condition holds good:

$a_1 \leq b, a_2 \leq b$ and $(\forall c): a_1 \leq c, a_2 \leq c$ imply $b \leq c$.

A function f is said to be *continuous* if f is monotonic and least upper bounds of chains in A are preserved, i.e., if the following condition holds good:

$\bigsqcup (f(a_1),...,f(a_n)) = f(\bigsqcup (a_1,...a_n))$

where $(a_1,.....,a_n)$ is an arbitrarily long chain

These concepts are standard in recursive function theory (e.g., see [2]).

3. Seitz's Weak Conditions: An Algebraic Formulation

In this section we first review Seitz's weak conditions and then formulate them in algebraic terms.

The weak conditions proposed by Seitz in [3] are a set of timing constraints on the allowable transitions of values on the data lines that form the inputs/outputs of a module. These constraints need to be preserved under composition of such modules. Intuitively, the weak conditions state that output lines lag behind the input lines and demand that the input lines be held steady until the output lines reach their final states. As figure 3–1 suggests, the value on a data line should go from being undefined ($\underline{\perp}$) to being defined (1 or 0) and go back to being undefined ($\underline{\perp}$) over a cycle. If we consider the vector consisting of the data values on the input lines over a cycle, it should change from being ($\underline{\perp}$,... $\underline{\perp}$) to (d,...d), where d is 0 or 1 and then go back to the vector ($\underline{\perp}$,...,$\underline{\perp}$). The output lines must go through the same transitions. For a formal statement of these conditions using primitives in temporal logic the reader is referred to [1]. The conditions as stated in [3] are recapitulated in figure 3–1.

We define a *cycle* to be the time period in which the input lines go from being completely undefined to being defined, and then back to being completely undefined. The first phase of the cycle is the time period in which the input vector goes from being completely undefined to being completely defined and the second phase of the cycle in which the input lines goes back from being completely defined to being completely undefined.

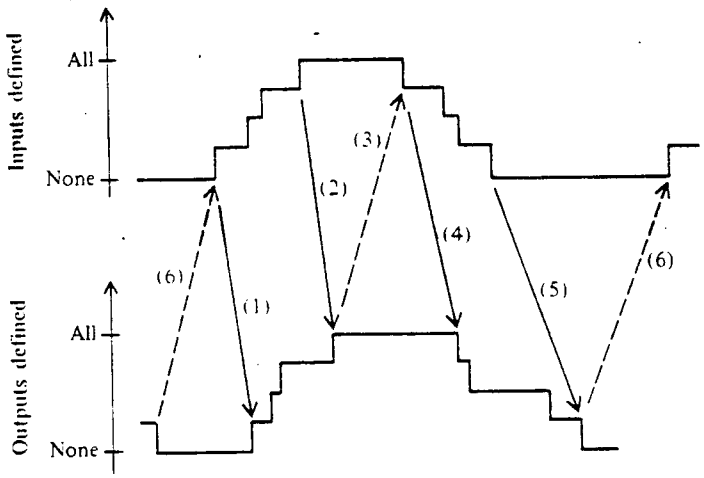
The constraints that embody the weak conditions can be broken down into two classes:

- Those that characterize the logic of the module.
- Those that characterize the module's interface with the environment and that therefore capture the behavior of the input and the output lines. The protocol that is being used (e.g., the request–acknowledge protocol) typically enforces these conditions.

Consider a module M defined by:

1. n input ports $(i_1,...,i_n)$ and
2. m output ports $(o_1,...,o_m)$

where the outputs are related to the inputs of the module through the functions $g_1,.....,g_m$ as follows:



The Seitz's conditions are as follows:
(1) Some inputs become defined ≤ some outputs become defined
(2) All inputs become defined ≤ all outputs become defined
(3) All outputs become defined ≤ some inputs become undefined
(4) Some inputs become undefined ≤ some outputs become undefined
(5) All inputs become undefined ≤ all outputs become undefined
(6) All outputs become undefined ≤ some inputs become defined

Figure 3–1: Weak Conditions

$o_1 = g_1 (i_{1,1},...,i_{1,k_1})$, where $(i_{1,1}..i_{1,k_1}) \subseteq (i_1...i_n)$

.
.
.

$o_m = g_m (i_{m,1},...,i_{m,k_m})$, where $(i_{m,1}..i_{m,k_m}) \subseteq (i_1...i_n)$

We define an input vector I for a module M as $<i_1,....,i_n>$, an output vector O for a module as $<o_1,...,o_m>$ and a combined I/O vector C for each module as $<i_1,...,i_n,o_1,....,o_m>$, where i_1 to i_n are the values on the n input lines to the module and o_1 to o_m are the values on the output lines at any time instance.

The weak conditions for a Module M can now be stated as follows:

Cond1: Given a sequence of I / O vectors $C_1,....,C_k$ at k time instances over a cycle, the elements of the sequence must satisfy the following relation,

$C_1 \leq C_2 \leq ....\leq D \geq .....\geq C_k$,

where $C_1 = \perp^{m+n}$
$C_k = \perp^{m+n}$,
$\perp^{m+n}$ is the vector $(\perp,...\perp)$ of $m+n$ elements,
D is the vector (d,...d) of $m+n$ elements, where d is either 1 or 0
and $\geq$ is the inverse of the partial order $\leq$.

Cond2: The functions $(g_i, i = i..m)$ must be total, monotonic and naturally–extended functions.

The first condition depends entirely on the environment and the input and output lines, whereas the second condition characterizes the logic of the module.

Given two modules which satisfy the two conditions, it can easily be shown that a composition of two modules with no feedback is closed under composition as follows:

Proposition 1: The condition *Cond1* is trivially satisfied under composition as, for any chain $\{S_i\}$ in domain D, and a chain $\{R_i\}$ in D', the sequence of tuples $\{<S_i,R_i>\}$ is a chain in the cross product D X D' of the domains D and D'.

Proposition 2: As composition of two total monotonic naturally–extended functions is again a total monotonic naturally–extended function [2], the condition *Cond2* is preserved under composition.

4. Sietz's Weak Conditions: A Temporal Logic formulation

We now briefly review some relevant definitions in temporal logic, state the formulation of weak conditions proposed in [1] and prove that the conditions *Cond1* and *Cond2* stated in the last section are equivalent to the temporal logic formulation of the same.

4.1. Temporal logic

Temporal logic is a modal logic, wherein the usual first order propositional calculus is upgraded with modal operators for reasoning about sequences over time. The modal operators used are usually from the set {○, <>, [], U} and when used with propositions, state properties about the future. The usual model used for temporal logic is a sequence of time instances, where the first element of the sequence is the *present* and subsequent elements are the *future*. Informally, the semantics of the modal operators are as follows:

○ p	the proposition p will be true in the next instance
<> p	p will be eventually true
[] p	p is true now and will remain true in all future time instances
p U q	either q is true now or p is true now and will remain true until q is true. Stated formally, $p\ U\ q = q \vee (p \wedge \bigcirc(p\ U\ q))$

Temporal logic can be used to reason about second order properties of parallel programs like termination, liveliness, freedom from deadlock etc.

In specifying some of the properties of asynchronous circuits, we will use some derived operators. Their definitions follow:

Operator	**Definition**
While	$p\ W\ q \equiv p\ U\ (\sim q)$

Precedes	$p\ P\ q \equiv (\sim q)\ U\ p$
Latched While	$p\ LW\ q \equiv p \supset p\ W\ q$
Entails	$p \sim\!> q \equiv p \supset <> q$

4.2. Weak conditions in Temporal Logic

In this section we will state the weak conditions in temporal logic [1] and argue that the conditions *Cond1* and *Cond2* are equivalent to the temporal logic specifications.

Throughout this section we will assume that we are given a module M, a set of input lines I, that has a set of output lines O together with a set of functions $\{g_i, i = 1..m\}$ such that,

$$o_1 = g_1 (i_{1,1},...,i_{1,k_1}), \text{ where } (i_{1,1}..i_{1,k_1}) \subseteq (i_1...i_n)$$

$$\vdots$$

$$o_m = g_m (i_{m,1},...,i_{m,k_m}), \text{ where } (i_{m,1}..i_{m,k_m}) \subseteq (i_1...i_n)$$

where $O = <o_i, i = 1..m>$ and $I = <i_j, j = 1..n>$ and the concatenated vector $C = <i_1...i_n, o_1...o_m>$.

Associated with the lines I and O, we define a function d, such that d(i) = true if the line i is defined (i.e., the value on the line i is either 0 or 1) and false when the value on the line is $\perp$.

We extend d to a set of lines, by defining

$$d(I) \equiv_{def} (\exists i \in I).(d(i))$$

and define a function D over a set of lines as

$$D(I) \equiv_{def} (\forall i \in I).(d(i))$$

In stating the weak conditions in temporal logic, we will state them in the form,

$$\alpha \supset [] w \text{ abbreviated as } \alpha \models w$$

where α is the set of initial conditions.

The initial condition α in all the temporal specification will be

$$\sim d(I) \ \Lambda \ \sim d(O).$$

The initial conditions correspond to a part of our condition *Cond1*. $\sim d(I)$ implies that $I = \perp^n$, and $\sim d(O)$ implies that $O = \perp^m$. As condition *Cond1* states that the vector C is initially $\perp^{m+n}$, the initial condition holds in our system.

In proving that our conditions are equivalent to the weak conditions as stated by Owicki, we need to prove that

Proposition 3: *Cond1* Λ *Cond2* $\supset$

	$\models \sim d(O)\ LW\ \sim d(I)$	(S1)
and	$\models \sim D(O)\ LW\ \sim D(I)$	(S2)
and	$\models D(I) \sim\!> D(O)$	(L2)
and	$\models D(I)\ LW\ \sim D(O)$	(E3)
and	$\models D(O)\ LW\ D(I)$	(S4)
and	$\models d(O)\ LW\ d(I)$	(S5)

and	$\models \sim d(I) \sim\!> \sim d(O)$	(L5)
and	$\models \sim d(I)$ LW $d(O)$	(E6)
and	$\models [D(I)$ P $\sim d(I)] \supset (\forall i \in I).[d(i)$ LW $\sim D(I)]$	(E7)
and	$\models [D(O)$ P $\sim d(O)] \supset (\forall o \in O).[d(o)$ LW $\sim D(O)]$	(S8)
and	$\models [\sim d(I)$ P $D(I)] \supset (\forall i \in I).[\sim d(i)$ LW $d(I)]$	(E9)
and	$\models [\sim d(O)$ P $D(O)] \supset (\forall o \in O).[\sim d(o)$ LW $d(O)]$	(S10)

We provide proofs for two of the conditions here, as all the others can be proved along similar lines.

Proof for Condition S1

Condition S1 states that

$Cond1 \wedge Cond2 \supset \models \sim d(O)$ LW $\sim d(I)$

i.e., $Cond1 \wedge Cond2 \supset \models \sim d(O) \supset \sim d(O)$ W $\sim d(I)$ (using the definition of LW)

i.e., $Cond1 \wedge Cond2 \supset \models \sim d(O) \supset \sim d(O)$ U $d(I)$ (using definition of W)

If the above were not true, it must the case that $\exists t : d(O) \wedge \sim d(I)$ at time t.

Hence, at time t, $I = (\perp \ldots \perp)$ and $g_j (\perp \ldots \perp) \neq \perp$, which contradicts the assumption *Cond2*. ■

Proof for condition E3

Condition E3 states that

$Cond1 \wedge Cond2 \supset \models D(I)$ LW $\sim D(O)$

The antecedent of the equation is

$D(I)$ LW $\sim D(O)$, which is equivalent to

$D(I) \supset D(I)$ W $\sim D(O)$, which is equivalent to

$D(I) \supset D(I)$ U $D(O)$. (*)

If the antecedent (*) is false, then

$\exists t : P \wedge \bigcirc(\sim D(I) \wedge \sim D(O))$ at time t,

where $P = D(I) \wedge \sim D(O)$

which implies that the state vectors at time t–1 (C_{t-1}), t (C_t) and t+1 (C_{t+1}) are

$C_{t-1} = (D(I), \sim D(O))$,

$C_t = (D(I), \sim D(O))$, and

$C_{t+1} = (\sim D(I), \sim D(O))$

which contradicts assumption *Cond1*,

Hence the condition E3 is true. ■

To complete the proof we have to prove that the implication in the other direction is also true,

i.e.

Proposition 4: S1 to S10 $\supset$ *Cond1* Λ *Cond2*

We first provide a proof for *Cond2*.

Consider an output line o_p. Without loss of generality we assume that the data value on the line o_p is a function of only one input line, say i_k. Thus we have that $o_p = g_p(i_k)$.

If the function g_p is not monotonic, this implies the existence of time instances j and j + 1 and of input and output vectors I_j, I_{j+1}, O_j, O_{j+1}, such that, when $I_j \leq I_{j+1}$, either $O_j \geq O_{j+1}$ or O_j and O_{j+1} are not comparable.

Given that we have a domain containing only three elements,

$O_j \geq O_{j+1} \supset o_p$ at j is 1 or 0 and o_p at j + 1 is $\perp$, without loss of generality. (**)

The statement (**) violates the condition S5.

If it is true that $I_j \leq I_{j+1}$ and O_j and O_{j+1} are not comparable, we must have that o_p at j is 1 and o_p at j + 1 is 0 or vice–versa. This violates our assumption that the transition between the data values 1 and 0 has to pass through $\perp$.

Hence the function g_p has to be monotonic, which proves our condition *Cond2*.

Before we proceed with the proof for *Cond1*, we wish to point out that the conditions E7, S8, E9 and S10 do not capture the requirement that the input or output values once defined, remain unmodified until all the other lines also become defined. The new conditions are as follows [5]:

$\sim d(I) \supset [\forall\, i \in I.\ d(i)\ LW\ \sim D(I)]\ U\ D(I)$ (E7')

$\sim d(I) \supset [\forall\, o \in O.\ d(o)\ LW\ \sim D(O)]\ U\ D(O)$ (S8')

$D(I) \supset [\forall\, i \in I.\ \sim d(i)\ LW\ d(I)]\ U\ \sim d(I)$ (E9')

$D(I) \supset [\forall\, o \in O.\ \sim d(o)\ LW\ d(O)]\ U\ \sim d(O)$ (S10')

The proof for *Cond1* now follows:

From E3 and S4 we obtain,

$D(I) \supset D(I)\ U\ \{D(I)\ and\ D(O)\}$ (E11)

$D(O) \supset D(O)\ U\ \{D(I)\ and\ D(O)\}$ (E11')

which says that once D(I) is true, it remains to be true until both D(I) and D(O) become true. The same is true for D(O) as well.

From E7', we obtain for any input line i,

$d(i) \supset d(i)\ U\ D(I)$.

From E11 and the definition of D(I), we get

$d(i) => d(i) \cup D(I)$ and $D(O)$ (E12)

Consider a sequence of time instances, such that $\sim d(I) \wedge \sim d(O)$ is true in the first instance, and both D(I) and D(O) are true in the last instance. From E12, it directly follows that the input line i, over this sequence, has values

$\perp, \ldots \perp, d(i), \ldots d(i)$ (E14)

Given the assumption that the transition in data values on any input or output line, from 0 to 1 or vice–versa has to pass through the value $\perp$, we can impose the following condition on the sequence of elements in E14

$\perp \leq \ldots \leq d(i) \leq d(i)$

Similarly from S8', we can obtain the condition for an output line o, as

$\perp \leq \ldots\ldots \leq d(o) \leq d(o)$

Taking the cross product of all input and output lines at all instances, we obtain from E12 and E13,

$\perp^{m+n} \leq C_1 \leq C_2 \leq \ldots D^{m+n}$,

which is part of our *Cond1*.

Similarly from E9' and S10', we obtain the other part of our *Cond2*,

$D^{m+n} \geq C_k \geq C_{k+1} \ldots \perp^{m+n}$,

which proves the proposition 4. ■

It follows from propositions 3 and 4 that the conjunct of the conditions S1 to S10' is equivalent to *Cond1* and *Cond2*. ■

5. Summary

We have presented two very intuitive (and algebraically simple to state) conditions that capture Seitz's weak conditions characterizing self timed circuits (and the 12 conditions discussed in [1]). In practice, specialized "protocols" for implementing asynchronous communication are designed that are intended to guarantee that the weak conditions are met e.g. the double rail/request–acknowledge protocol. Thus, the average designer does not have to be cognizant of the conditions themselves, but only needs to know one or more protocols that provide "correct" implementations of the conditions. However, the conditions to be obeyed may in general be viewed as "specifications" that any newly proposed protocols must satisfy. Thus, in addition to being intuitive, it is hoped that the algebraic statement of the conditions to be met will facilitate such proofs in the overall framework for design that has been developed in [4].

References

[1] Y. Malachi and S.S.Owicki, *Temporal Specifications of Self–Timed Systems*, pp. 203–212, Proceedings of the 1981 CMU Conference on VLSI Systems and Computations, Computer Science Department, Carnegie–Mellon University, (October 1981).

[2] Manna, Z, *Mathematical Theory of Computation,* (McGraw--Hill, 1974).

[3] Charles Seitz, System Timing in: Mead C. and Conway L., Ed., *Introduction to VLSI systems*, (Addison–Wesley, Reading MA, 1980), pp. 218–262.

[4] Subrahmanyam, P.A., An Algebraic Basis for VLSI Design, Draft of a Research Monograph, April 1982. Available from the Department of Computer Science, University of Utah.

[5] Yoni Malachi, Personal Communication .